Rakesh C. Singh

Prototype Development of SmartPDA (Personal Digital Assistant) Using BLE

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Thesis

25Th August 2015

Abstract

Author(s)	Rakesh C. Singh
Title	Prototype Development of SmartPDA (Personal Digital Assistant) Using BLE
Number of Pages	72 pages
Date	25Th August 2015
Degree	Master of Engineering
Degree Programme	Information Technology
Specialization	Embedded Systems and Engineering
Instructor	Kari Järvi, Principal Lecturer

This thesis introduces and discusses smart devices and Android application development with a special focus on the use of external sensors and communication as part of an application.

The study first introduces short-range wireless technology, smart devices and prototype development in general, going through the most common components in Android applications development and then takes a closer inspection on the development of a prototype with an external sensor and communication channels.

The aim of the study was to create a prototype with low energy requirement, sensors connection with mobile platform and relevant supporting application. The sensors data are transferred via Bluetooth Low Energy (BLE) technology between smart device application and the developed prototype device. Therefore, an introduction to this BLE technology is also included.

The selection of sensors and communication channels was the main objective of the project. Respective platforms were selected to develop the prototype. The application is targeted at the latest Android version. The major obstacle in the development process was lack of proper sources and guides which increased the development time significantly. Minor challenges originated from the integrated development environment which reportedly has some issues with Android development.

The smart device prototype was created to a point where all the functionalities for using the sensor and saving data were completed. The shape and usage of the final product were left for future development.

However, a simple user interface was created for testing purposes. The application in itself offers little to none monetary value. In the future, a number of components will be added and shared as open source for the benefit of other developers. Adding more functionalities and finalizing the user interface is left for future evaluation cycles.

The study concludes on a wearable prototype/product for initiating an emergency call for medical or security help without operating a smartphone via the application.

Keywords	Embedded System and Engineering, Smart Device, BLE, Sensors

Table of Contents

Abbreviations

3G	3rd Generation
AP	Access Point
API	Application Programming Interface
BLE	Bluetooth Low Energy
CIS	Continuous Integration System
dBm	DeciBel relative to one milliWatt
GHz	GigaHertz
GPIO	General Purpose Input Output
IEEE	Institute of Electrical and Electronics Engineering
ISM	Industrial, Scientific and Medical Radio Band
JSON	JavaScript Object Notation
Kbps	Kilobits per Second
LE	Low Energy
LED	Light-emitting diode
LSI	Large Scale Integration
m/s	Meters per Second
MHz	MegaHertz
MVC	Model View Controller
NFC	Near Field Communication
OFDM	Orthogonal Frequency-Division Multiplexing
PDA	Personal Digital Assistant
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
SIG	Special Interest Group
SMS	Short Messaging Service
UART	Universal Asynchronous Receiver Transmitter

1. Introduction

Human behavior and lifestyle is changing rapidly in modern society. People are using technology in every facet of their life. With the advancement of technology, humans are becoming more and more dependent on the technologies. This dependence raised the demand of more smart devices in the market as people were getting inseparable from their smart devices and it created big opportunity for these devices to succeed. A smart device is an electronic device connected to other devices or networks via different wireless protocols such as Bluetooth, NFC, WiFi, 3G etc., and can operate to some extent interactively or autonomously. Smart devices are more commonly operated and controlled by a smartphone which is a smart device as well. The availability of advanced and low cost technological options make smartphone capabilities and options unlimited. Combination of smartphone and smart device solutions provide limitless possibilities that could benefit mankind in all aspects.

Smartphones are widely accepted in our life and ecosystem. A smartphone is a combination of traditional mobile phone and applications to provide information on one simple tap. Millions of applications are available across the platform to serve similar features such as email, calendar, clock, browsers, chat, social networking, health monitoring etc.

The combination of smartphone and smart devices offers unmatched services and enhance the user experience. The synced applications on smartphone with smart device such as smartwatch, smart band or smart keychain provide hassle free services. Every

smart device has a different purpose and has limitations. For instance, a smartwatch can sync phone activities (call operation, notification updated) with day running battery life, smart band monitor your heart rate/pulse/all movement activities and sync to an application running on the smartphone but limiting phone call via band, smart keychain alerts user in case the keys are about to be lost or left unattended via an application running on the smartphone. These smart devices are compositions of sensors such as temperature, heart beat reader, pulse or oxygen level, accelerometer, gyrometer, magnetometer and communication medium. These sensors have different usage in different conditions and widely distributed among the product with dependency to the smartphone platform. At the same time, the usability of such devices is predefined with no possibilities for further enhancement.

Overall, the benefit from the existing system is limited and provides one way usage of informing the smartphone of the activity on a smart device. The capabilities of smart devices are unexplored in cases of medical emergency or personal security. [2] These smart devices can be used in a more innovative way to provide round the clock usability to anyone for accessing or requesting medical help, personal security via smartphone without even touching or operating smartphone. These solutions can be more beneficial to physically disabled people, patients and any person in need of help in any type of situation. In the form of wearable, these device can provide extended flexibilities to use in every moment of life and increase the experience of smart technologies. [1]

The goal of this project was to develop a Smart Personal Digital Assistant (SmartPDA) prototype which is a combination of a smart device (hardware) and a supporting application (software) across all existing platforms. The prototype focuses on

low power requirement and fulfills the low manufacturing cost model. The prototype is a new set of smart device in the form of wearable and provides a very easy way to make emergency calls in all situation without touching or any interaction to a smartphone within the range. It also provides the options to draw gestures in air for operating a smartphone within the range. The power consumption of the device will be minimal. The wearable device is expected to run on a coin size battery for a long period.

The study also focuses on the communication medium between the smart device and smartphone as it has a significant role in power consumption. Smart devices are supposed to work for long periods of time with very little power consumption. Bluetooth specific BLE is the most suitable match for the smart device communication channel. BLE gives an idle 10-20 meter radius range and can last for year on a coin size battery. [9]

The prototype is expected to be ready for mass production as a wearable smart device and supporting application across all the available mobile platforms. The prototype is a complete package of the hardware and respectively supporting software.

This study includes the full coverage of the project. The study focuses more on the development process involved. During the study respective analysis was carried out to meet the requirements, such as the selection of the short-range wireless communication, hardware cost, availability and usage. It also focuses on the selection of the development environment for application development. At end of the study a prototype of final product is introduced. The study concludes by the evaluation of the proposed prototype and the final sample product in the format of a wearable smart device.

1.1 Short-range Wireless Technology

A short-range radio is a highly integrated transceiver usually on a single chip used to implement a wide range of wireless data applications. Most single-chip transceivers operate in the industrial-scientific-medical (ISM) bands set aside for unlicensed wireless data applications. Popular U.S. frequencies include 315, 433, and 902 to 928 MHz, as well as 2.4 to 2.5 GHz and 5.7 to 5.8 GHz. In Europe, 868 MHz is popular.

1.1.1 Infrared

Infrared technology is the one of the most mature technologies. It has become extremely popular as an infrared data transmission technique for home computers, input devices such as wireless keyboards and mice. Figure 1 shows the IRDA logo printed on the body of IR capable devices.

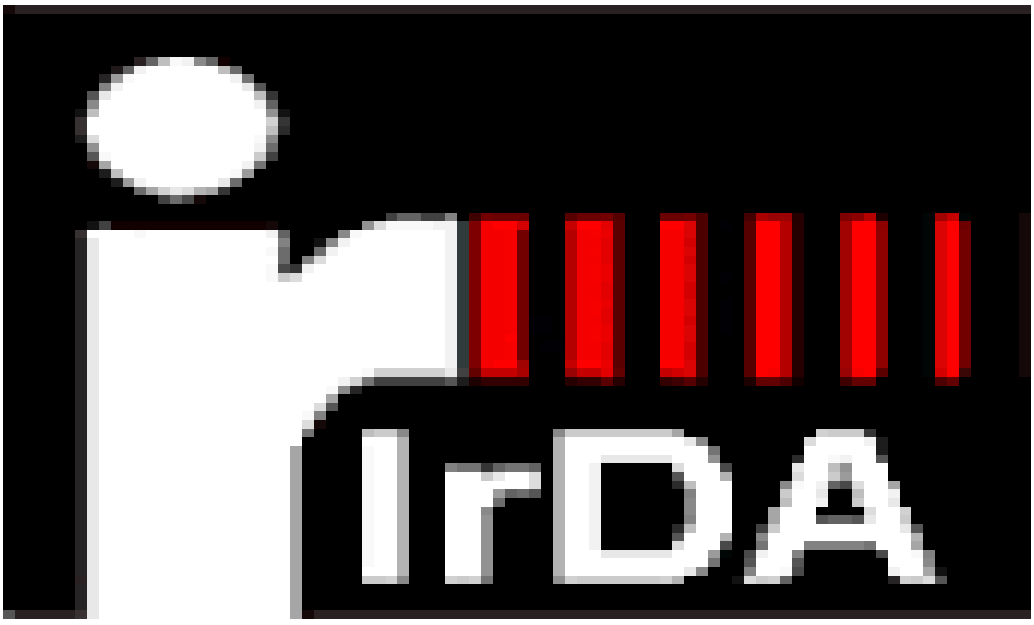


Figure 1: IRDA Logo

Many computers come with IrDA ports which allow data to be transferred from one computer to another computer or mobile phones. They can exchange information between

any computer and printer without any cable connection for printing. IrDA is made using an embedded LED which is in general a light electronic diode component, emitting infrared waves with very high brightness which have very narrow transmit beam. The element has the most important role in the transmission process. In the simplest link of point-to-point data transmission, most of the infrared light is directed from a source LED or semiconductor laser diode LD to photodiode which is called a detector. A transmitter converts an electrical signal to an optical signal. After having captured optical light, a detector converts optical power into electrical current. The best advantage is the capability to carry a high bandwidth up to 4 Mbit/s. Moreover, the last infrared version supports data transmission between devices up to 16 Mbit/s. The high speed of the transmission is achieved by using a protocol called Very Fast Infrared (VFIR). [19]

The main disadvantage is the data transmission possibilities requirement of devices to be in line of sight and the nature of infrared energy. It is one form of light which can be easily blocked and it cannot pass through solid objects. The infrared wavelength signals range starts at the end of the microwave band and ends at the beginning of visible light in the available spectrum. The working interval of infrared wavelength is defined from 750 nm to 1 mm. [20]

1.1.2 ZigBee

An interactive standard ZigBee was developed for applications that require low data rate wireless networking, low battery power consumption and low running costs. However, the standard provides great flexibility compared to the other network channels, with reliable and secure communication. Figure 2 illustrates the logo of ZigBee specified by ZigBee Alliance.

Figure 2: ZigBee Logo

Most of the modern wireless communications were developed specially to meet the highest technical characteristics in order to achieve higher data transmission along with longer range of distance to communicate. In fact elements such as sensors and controls of an active network do not really need to have high bandwidth but definitely need very low battery consumption in order to save battery power. [22] A ZigBee 802.15.4 network is designed to communicate at 1 mW of radio frequency power. It is the standard technology which is addressed to applications such as remote monitoring, control and sensor network functions for short distance communication.

1.1.3 NFC (Near Field Communication)

NFC is a form of contactless communication between devices like smartphones or tablets. Contactless communication allows a user to wave the smartphone over a NFC compatible device to send information without needing to touch the devices together or go through multiple steps for setting up a connection. Figure 3a illustrates the logo of NFC generally printed on NFC accessible devices.

Figure 3 a: NFC Logo

Figure 3 b: Smartphone with NFC

NFC is a set of short-range wireless technologies, typically requiring a distance of 10 cm or less. NFC operates at 13.56 MHz on ISO/IEC 18000-3 air interface and at rates ranging from 106 kbit/s to 424 kbit/s. NFC always involves an initiator and target; the initiator actively generates an RF field that can power a passive target. This enables NFC targets to take very simple form factors such as tags, stickers, key fobs, or cards that do not require batteries. NFC peer-to-peer communication is possible, provided both devices are powered. Figure 3b illustrates the contactless communication between smartphone

and NFG tag. Figure 3b also demonstrates the modes of communication. In general there are two modes of communication explained below. [21]

Passive communication mode: The initiator device provides a carrier field and the target device answers by modulating the existing field. In this mode, the target device may draw its operating power from the initiator-provided electromagnetic field, thus making the target device a transponder.

Active communication mode: Both initiator and target device communicate by alternately generating their own fields. A device deactivates its RF field while it is waiting for data. In this mode, both devices typically have power supplies.

Figure 3b demonstrates the passive communication mode where a smartphone is the device with power generation capabilities and tag with information to transmit. These types of products are very useful for predefined information to be supplied or transmitted e.g. in a museum the information of a statue such as build year, maker, country etc. Depending on the structure and requirement of very short range with very small need of information, NFC and its mode is defined and implemented.

1.1.4 Bluetooth

Bluetooth technology is a wireless communications technology that is simple, secure and exchanging data over short distances (using short-wavelength UHF radio waves in the ISM band from 2.4 to 2.485 GHz, using a spread spectrum, frequency hopping, full-duplex signal at a nominal rate of 1600 hops/sec) from fixed and mobile devices, and building personal area networks (PANs). The 2.4 GHz ISM band is available and unlicensed in most countries. [7]

Figure 5 shows the Bluetooth logo on top standardized by Bluetooth SIG. It also demonstrates the Bluetooth usage in our daily life. Due to short range for the signal

transmission, Bluetooth can create as many as PAN on demand bases and it does not interfere the other PAN due to its technology implementation. Bluetooth is ideal for the personal area network for a short range with a limited number of devices.

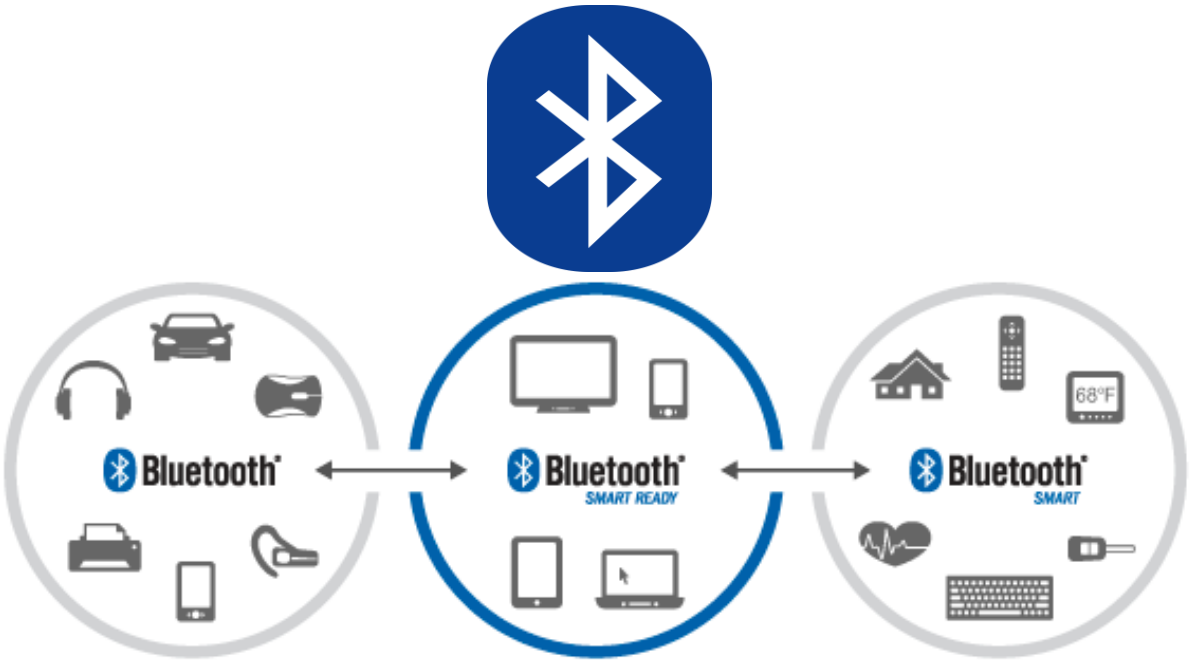


Figure 4: Bluetooth Logo on Top and Usages of Bluetooth (Classics, Smart and Smart Ready)

The key features of Bluetooth technology are ubiquitousness, low power and low cost. The Bluetooth Specification defines a uniform structure for a wide range of devices to connect and communicate with each other. Connections between Bluetooth enabled electronic devices allow these devices to communicate wirelessly through short-range, ad hoc networks known as piconets. Piconets are established dynamically and automatically as Bluetooth enabled devices enter and leave radio proximity.

1.1.5 Bluetooth Low Energy

Bluetooth low energy (Bluetooth LE, BLE, marketed as Bluetooth Smart) is a wireless personal area network technology designed and marketed by the Bluetooth Special Interest Group aimed at novel applications in the healthcare, fitness, beacons, security, and home entertainment industries. Originally introduced under the name Wibree by Nokia in 2006. Table 1 introduces a summary of short range wireless communication technologies

Communication Technologies

	NFC	RFID	Blue-tooth®	Blue-tooth® LE	ANT	Proprietery (Sub-GHz & 2.4 GHz)	Wi-Fi®	ZigBee®	Z-wave	KNX	Wireless HART	6LoWPAN	WiMAX	2.5–3.5 G
Network	PAN	PAN	PAN	PAN	PAN	LAN	LAN	LAN	LAN	LAN	LAN	LAN	MAN	WAN
Topology	P2P	P2P	Star	Star	P2P, Star, Tree Mesh	Star, Mesh	Star	Mesh, Star, Tree	Mesh	Mesh, Star, Tree	Mesh, Star	Mesh, Star	Mesh	Mesh
Power	Very Low	Very Low	Low	Very Low	Very Low	Very Low to Low	Low-High	Very Low	Very Low	Very Low	Very Low	Very Low	High	High
Speed	400 Kbs	400 Kbs	700 kbs	1 Mbs	1 Mbs	250 kbs	11-100 Mbs	250 kbs	40 Kbs	1.2 Kbps	250 kbs	250 Kbs	11-100 Mbs	1.8-7.2 Mbs
Range	<10 cm	<3 m	<30 m	5-10 m	1-30 m	10-70 m	4-20 m	10-300 m	30 m	800 m	200 m	800 m (Sub-GHz)	50 km	Cellular network
Application	Pay, get access, share, initiate service, easy setup	Item tracking	Network for data exchange, headset	Health and fitness	Sports and fitness	Point to point connectivity	Internet, multimedia	Sensor networks, building and industrial automation	Residential lighting and automation	Building automation	Industrial sensing networks	Senor networks, building and industrial automation	Metro area broadband Internet connectivity	Cellular phones and telemetry
Cost Adder	Low	Low	Low	Low	Low	Medium	Medium	Medium	Low	Medium	Medium	Medium	High	High

Table 1: Summary of short range wireless communication Technologies

The advantages of short-range wireless technology over each other in terms of coverage range and power requirement can be easily identified with the help of Table 1. This information can also be useful for drawback comparisons for new product designing based on short-range wireless technologies.

1.2 Smart Devices

Smart devices are designed to support a variety of form factors, a range of properties pertaining to ubiquitous computing and to be used in three main system environments: physical world, human cantered environments and distributed computing environments.

1.2.1 Smart Phone

Smart phones is a category of mobile devices that provide advanced capabilities beyond a typical mobile phone. Smart phones are based on advanced mobile operating system (OS), that provides a standardized interface and platform for application. Typically it is a combination of cell phone features with other popular mobile devices, such as personal digital assistant (PDA), media player and GPS navigation unit.

Compared to standard phones, smartphones usually have larger displays, powerful processors, reasonable RAM and capabilities to run third-party application. The hardware composition slightly varies among the manufacturer but in general a camera, motion sensors and communication channels are common. During the last few years high-speed mobile broadband 4G LTE capabilities have become standards for this category. The current smartphone home screen UIs across all platforms shown in Figure 5.

Android OS platform based phones have major share of the mobile phone market followed by the iOS platform. Both platforms gain the popularity is last couple of years. Both platforms provides wide range of flexibilities for user and developer. Both platforms provides API’s, IDE and many more supports facilities to the developer for developing application for the user on their platform.

(Android 5.0 Lollipop)

Apple iOS 8.3

Windows Phone 8.1

Figure 5: Smartphone's Home screen from different manufacturers.

The left home screen is from an Android OS running phone. Very simple UI design and few main functionality keys on bottom of the screen. The home screen have default application used mostly by any user but completely customizable.

The center home screen in Figure 5 is a very popular iOS platform from Apple. Home screen UI is very simple in design and all application are at the home screen. The home screen is completely customizable as the user wishes. The home screen has some standards for application icons which are followed by all application.

The right home screen is from the Windows Phone platform. The UI is of very modern design and a little complex for a first time user. The Windows Phones have a different structure but the basic functionalities are the same as in the other two platform.

In general every smart phone has common functionalities and features such as phone call, SMS, email, Phone book, Browser etc. and some advanced features on individual platforms. One good example is iOS Siri which is a completely voice command based feature.

1.2.2 Smartwatch

Smartwatch is a wristwatch with composition of sensors, communication medium and computerization capabilities. The early version of a digital watch with small computing power works independently for case specific such as calculator, meeting reminder. Today's smartwatch collects information from internal or external sensors, retrieves data from other instruments or computers and have capabilities to even control them through wireless technologies like Bluetooth, Wi-Fi, and GPS. Figure 6 shows a wristwatch with a calculator.

Figure 6: Wristwatch

Sport watch is another category within smartwatch but mainly focuses on the activity done by human body. It is meant for training, diving and outdoor sports with sensors such as heart rate, pulse, GPS tracking.

The latest versions of most smartwatch models manufactured today are completely functional as standalone products, many of the devices that are manufactured now are required to be paired with a mobile phone running the same operating system, and this

allows the watch to run not only as a watch but a remote to the phone. This allows the device to alert the user to communication data such as calls, SMS messages, emails, and calendar invites. Figure 7 provides comparisons between smartwatches. General features among all are:

a. Anti-lost alert
b. Time display
c. Call vibration
d. Caller ID
e. Answer call
f. Micro USB input port





	Apple **Apple Watch**	Asus **ZenWatch**	Sony **Smartwatch 3**	LG **G Watch R**	Samsung **Galaxy Gear S**	Motorola **Moto 360**
Processor	Unknown	1.2 Ghz	1.2 Ghz	1.2 Ghz	1 Ghz dual-core	1 Ghz
Storage	Unknown	4GB	4GB	4GB	4GB	4GB
RAM	Unknown	512MB	512MB	512MB	512MB	512MB
Heart rate monitor	Yes	Yes	Yes	Yes	Yes	Yes
Touchscreen	Yes	Yes	Yes	Yes	Yes	Yes
NFC	Yes	No	Yes	No	No	No
GPS	No (pairs with iPhone)	No	Yes	No	Yes	No
SIM card slot	Unknown	No	No	No	Yes	No
Connectivity	Unknown	Micro USB	Micro USB	Micro USB	Micro USB	Micro USB
OS	iOS	Android Wear	Android Wear	Android Wear	Tizen	Android Wear
Camera	No	No	No	No	No	No
Bluetooth	4.0	4.0	4.0	4.0	4.1	4.0

Figure 7: Short comparison

The features or capabilities of a smartwatch are limited for the time being but it promises more extended usage in future. As can be seen in Figure 7 the hardware configurations are the same among the different manufactures. All have some unique add-on compared to its peer. These smartwatches have prices in higher range and provide

predefined functionalities such as syncing smartphone activities or updating user movement to smartphone application.

1.3 Prototype Development

Prototype development is the mother on any invention and provides an opportunity to test any idea in reality. This type of development also provide the fastest way to collect feedback and make the changes to the product depending on the feedback. The process of prototyping involves the following steps:

a. **Identify basic requirements:** Determine basic requirements including the input and output information desired. Details, such as security, can typically be ignored. [3]
b. **Develop Initial Prototype:** The initial prototype is developed that includes only user interfaces.
c. **Review:** The customers, including end-users, examine the prototype and provide feedback on additions or changes.
d. **Revise and Enhance the Prototype:** Using the feedback both the specifications and the prototype can be improved. Negotiation about what is within the scope of the contract/product may be necessary. If changes are introduced then a repeat of steps #3 and #4 may be needed. Figure 10 depicts the prototype development steps.

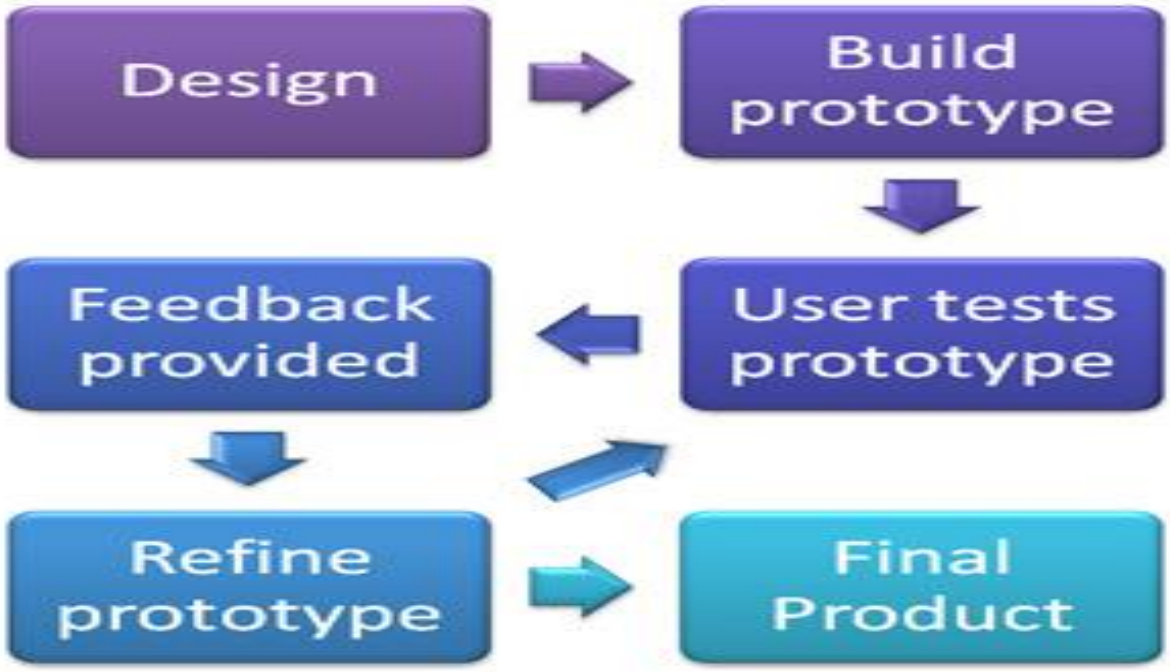


Figure 10: Prototype development

The three types of prototype development categories are:

Visual prototype: This type of prototype design demonstrates the overall shape and size of the product and does not usually feature any working parts. The materials will not be the materials that would be used if the product were to be mass produced. This type of product prototype can either be left as the raw material or painted to represent the finished product. [3]

Proof-of-concept prototype: This type of prototype product demonstrates the key functionality and resolves the main technical aspects of the design. It is not intended to look like the final product and will make use of existing 'off-the-shelf' components, where possible. A proof of concept prototype is unlikely to be made from production grade materials. [3]

Presentation prototype: This type of product prototype demonstrates the key functionality and also provides a representation of how the mass-produced product would appear. This type of prototype will largely be built from bespoke parts; however 'off-the-shelf' parts will still be used, where appropriate. It is likely that materials close to the production grade will be used, however, in situations where this would not be economically viable for a one-off prototype, substitutes may be used. [3]

1.4 Summary

This chapter described the existing structure of the smart devices and prototype development. It also covered parts of available short-range wireless communication channels and their dependency. The following Chapter 2 focuses on the development environment and technology involved in this project. It also presents a detailed structure of Bluetooth technology, Mobile Application IDE and PCB designing. Chapter 3 describes the Low energy technology, protocol and usage structure. Chapter 4 presents the prototype development and focuses on low level hardware designing. Also, the text briefly elaborates on the GAP/GATT protocol and usage structure. Chapter 5 describes Prototype integration and testing. The sections in this chapter are more focused on application development and testing. Chapter 6 shows the case usage of the prototype as a final product.

2. Development Environment and Technology

This chapter describes more in detail the project related to the development environment and technologies used to achieve the goal. At end of this chapter the complete development structure takes shape and provides a good starting point for further development.

2.1 Bluetooth

This section describes Bluetooth and its history. It also focuses on the overall protocol and structure of Bluetooth. The core of this section is methods and technologies behind Bluetooth to function successfully, such as bandwidth, frequency rate etc.

2.1.1 Bluetooth SIG

Bluetooth as a name of a technology has a huge historical background from the 10th of century Danish King Harald Blåtand or Harald Bluetooth in English. More than a thousand years after the death of the Danish king, the company Ericsson Mobile Communications established a research group to study the possibility of using radio waves to provide interaction of different devices at short distances. When the Swedish company was satisfied with the results of the research work, the representatives of Ericsson applied for cooperation with other firms. Before, in 1998 five companies: Nokia Corporation, International Business Machines Corporation (IBM), Intel Corporation, Toshiba and Ericsson Corporations had organized a group of Special Interest Group (SIG), which was actively engaged in the development of the Bluetooth standard. Nowadays Bluetooth is a well-known trademark of Bluetooth Special Interest Group, the group of leaders in the telecommunication systems, computing technologies, consumer electronics, automotive

and network industries, actively working on the development of standard Bluetooth and bringing it to word market. The technology is already has been accepted by over 2100 companies around the world. [8]

2.1.2 Bluetooth Wireless Technology

Bluetooth technology is the world leader and the most successful short range wireless communications technology to support both voice and data transmission to create a personal area network. Many chips are embedded into a huge number of technical devices beginning from cellular phones, headsets and stereo headphones, through to medical devices, portable media players and games consoles. Figure 12 shows the smallest size of BLE module (15mmx18mm). [12]

Figure 12: BLE Module

At beginning the main idea for Bluetooth technique was that it has to have a universal radio interface is connecting to different devices communicating to each other and provide low cost wireless communication that uses radio technology. Bluetooth devices are a very good solution for applications especially when designers need low power consumption, excellent performance, and minimum size.

Bluetooth connects a huge number of devices allowing them to use a high-speed, low-power wireless link technology. It is designed to automatically connect portable

equipment together when one Bluetooth device is on a work distance to another device with the same standard. The network protocols are able to provide secure, reliable connectivity of transferring user voice, required data with a very flexible network topology. Totally different from infrared technology, the Bluetooth the device does do not have to be in visual line of sight. The most notable feature is that the radio chips are very small and can fit in any electronic device. In Bluetooth communication a user does not need to be involved the in process of establishing communication. Bluetooth devices are enabled dynamically every time constantly sending address information and establishing a network between the devices.

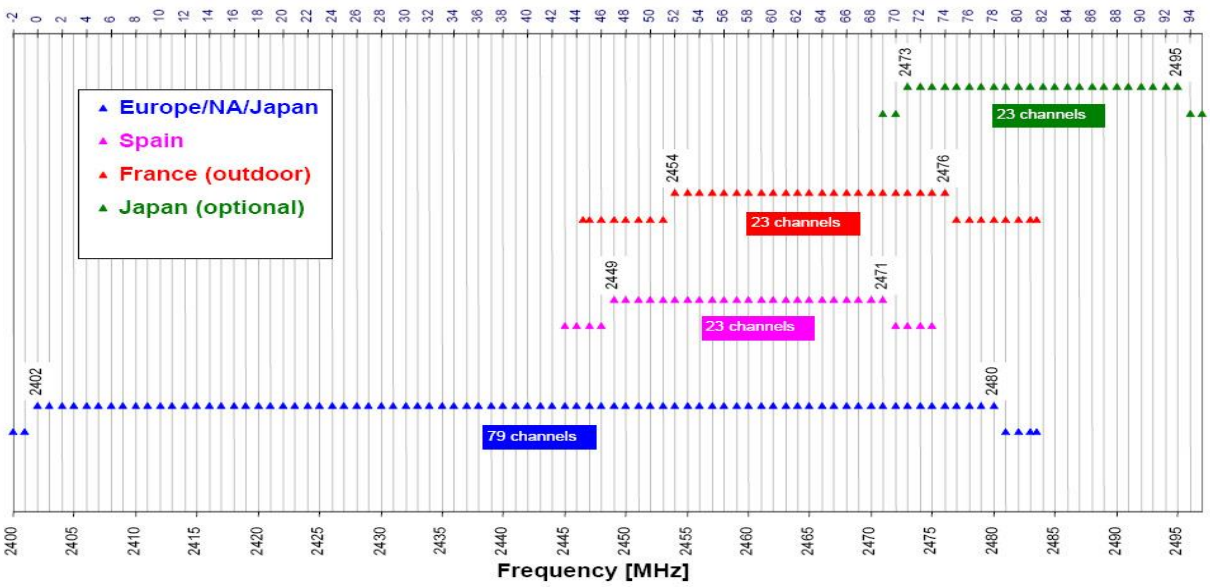
Figure 11: Bluetooth connection

Figure 11 illustrates Bluetooth connection and supported devices. This is also a very good example of Bluetooth as a connection medium among commonly used gadgets. A single Bluetooth device can handle and serve 7 actively connected devices at a time but can provide services to an unlimited number of devices. This works in FCFS (First Come First Serve) style, when an active device is inactive or out of range the services switch to the next available device in the range with contingency of the maximum number of connected devices.

2.1.3 Radio Frequency Hopping

The work frequency for Bluetooth devices operates globally around the world. This is the license free frequency band at 2.45 GHz open to any radio system. The frequency range differs according to the rules of a country where s Bluetooth device is used. For United States, Japan and Europe the unlicensed range has certain limits from 2,400 MHz to 2, 483.5 MHz if the transmitting power exceeds 0 dBm. However, some European countries still allow a part of the whole spectrum, only 23 channels are available for transmitting in France, Japan and Spain. This is illustrated in Figure 13. [11]

Figure 13: Frequency usage



Bluetooth uses frequency-hopping spread-spectrum technology (FHSS). The frequency hop technique divides the whole frequency spectrum into several hop channels with a nominal bandwidth 1 MHz for each channel. During a connection Bluetooth transceivers use all 79 channels and rapidly hop from one channel to another in a random manner across all the channels. Every channel is divided into time segments, slots with a duration of 625 µsec in length. All slots in the range are numbered from 0 to 2^{27} -1 and in

accordance with the clock master piconet. The transceiver is using only one channel at a time. When one device is connected to another Bluetooth device, it hops with a hop rate of 1600 hops per second lingering with a residence time of 625 µsec on any randomly given frequency channel. A device cannot be operating more than 0.4 second within any 20 second period for frequency hopping systems operating in the 902–928 MHz band. In the 2400 - 2483.5 MHz band the average time of a channel is not more than 0.4 seconds within a period of 0.4 seconds multiplied by the number of employed channels. The restrictions are applied by FCC regulations to limit interference in the unlicensed industrial, scientific and medical (ISM) band. Figure 13 elaborates the channels used in a given country and Figure 14 shows frequency hopping. [11]

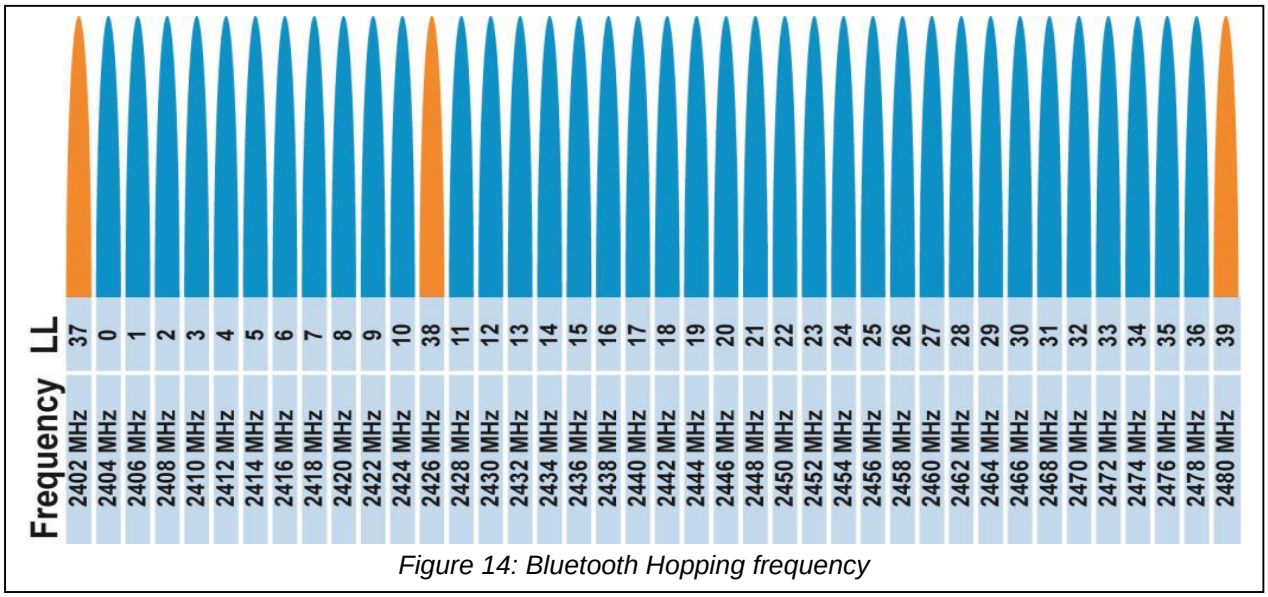


Figure 14: Bluetooth Hopping frequency

Figure 14 illustrates the hopping and selection of the frequency within the range of 2.4 GHs. The band is divided into 40 channels, some of them are reserved to initiate primary setup and discovery but the rest of the channels are used randomly as Bluetooth stack specific. This makes it very hard to interfere other devices within the same range to cross talk or create any type of signals collision as the hopping takes place 1600 times per sec.

2.1.4 Orthogonal Frequency Division Multiplexing

OFDM or Orthogonal Frequency Division Multiplexing is a special type of multicarrier modulation. It is a kind of a replacement of a single high frequency carrier by multiple sub carriers that are used for one transmission link between two stations. The transmitted data stream is distributed over a large number of channels and the data is sent over all the channels simultaneously. High speed is the advantage of the simultaneous data transmission even if the transmission rate on every channel is not so high. Independent to each other, sub channels have very tight frequency intervals between them in order do not make frequency interference.

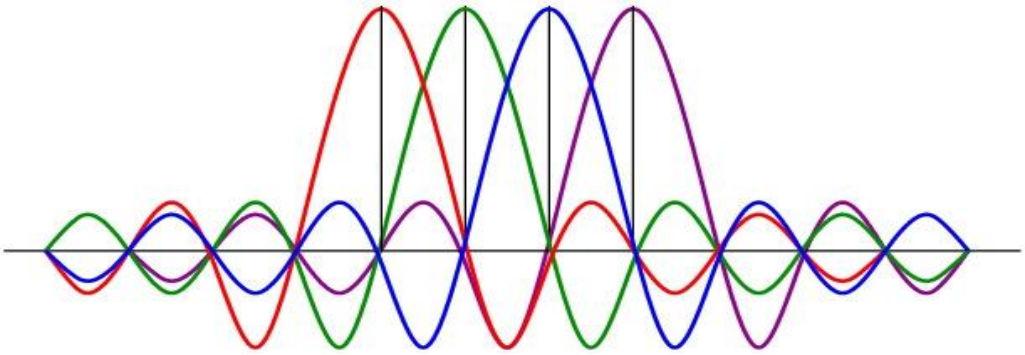


Figure 15: Frequency overlapping

As shown in Figure 15, sub channels overlap each other because of the form of the carriers are orthogonal makes frequency spectrum in more efficient way. [11]

2.1.5 Bluetooth Network Topology

The typical Bluetooth network topology is based on two main net concepts, “Piconet” and “Scatternet”. The Bluetooth protocol can support connection between one or more Bluetooth devices by simply using concepts of two networks. When piconet includes only two connected devices, the net refers to a point-to-point connection scheme or other specific scheme as point-to-multipoint net with more than one slave device connected together. One or more devices create a piconet if they are synchronized on the

same physical channel. In the Bluetooth network two types of connection are possible: synchronized or asynchronous connection.

The Synchronous Connection-Oriented (SCO) link is used for a point-to-point connection and more orientated to establish connection which is usually used to transmit voice packets or mixed voice and data packet. The Asynchronous Connectionless (ACL) link supports point-to-multipoint connections without establishing a connection between devices, and supports symmetric and asymmetric connections point-to-multipoint packet-switching, which are commonly used to transmit data. All devices in a piconet hop together. Piconets are not synchronized to each other and every piconet has their own sequence of frequency-hopping. In a typical piconet only one of the devices works as a master and the rest of them act as slaves.

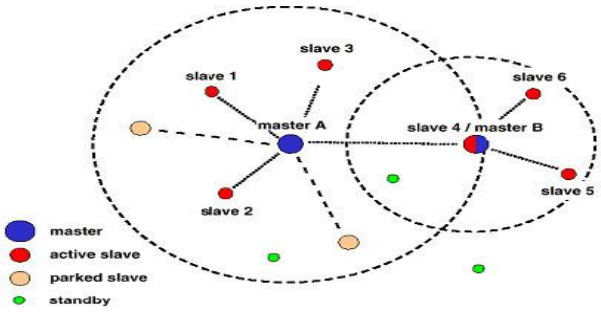


Figure 16: Master-Slave compositions

Figure 16 illustrates the master and slave structure of the Bluetooth connected device. There is no limit in the number of slaves. The master can connect 7 devices which is the maximum number of active slaves connected simultaneously. [11] If there are more than seven slaves, the rest of the slaves must be “parked”. The number of parked slaves can be up to 256 per piconet. Every parked device gets directly an 8-bit Parked Member address (PM_ADDR). Any parked device is synchronized by master clock, which can quickly change their current state to the active state and get 3-bits Active Member address (AM_ADDR).

2.2 Mobile Application Development

This section of the chapter is mainly a coverage of mobile application development. A detailed description of the development IDE available for different platforms is provided. The chapter also describes more in detail the Printed Circuit Board and related software.

2.2.1 Android Studio IDE

Android Studio is an integrated development environment (IDE) for developing application for the Android platform. The core of Android Studio is an intelligent code editor capable of advanced code completion, refactoring, and code analysis which is based on IntelliJ IDEA. Android Studio comes preconfigured with an optimized emulator image. The updated and streamlined Virtual Device Manager provides predefined device profiles for common Android devices.

Android Studio provides a convenient platform for building applications for Android phones, tablets or Android Wear for all kind of developer. The new Android Project View makes it easier to manage app projects and resources during development. Some of features are as follow:

a. Flexible Gradle-based build system
b. Build variants and multiple apk file generation
c. Code templates to help you build common app features
d. Rich layout editor with support for drag and drop theme editing
e. lint tools to catch performance, usability, version compatibility, and other problems
f. ProGuard and app-signing capabilities
g. Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

h. And much more ...

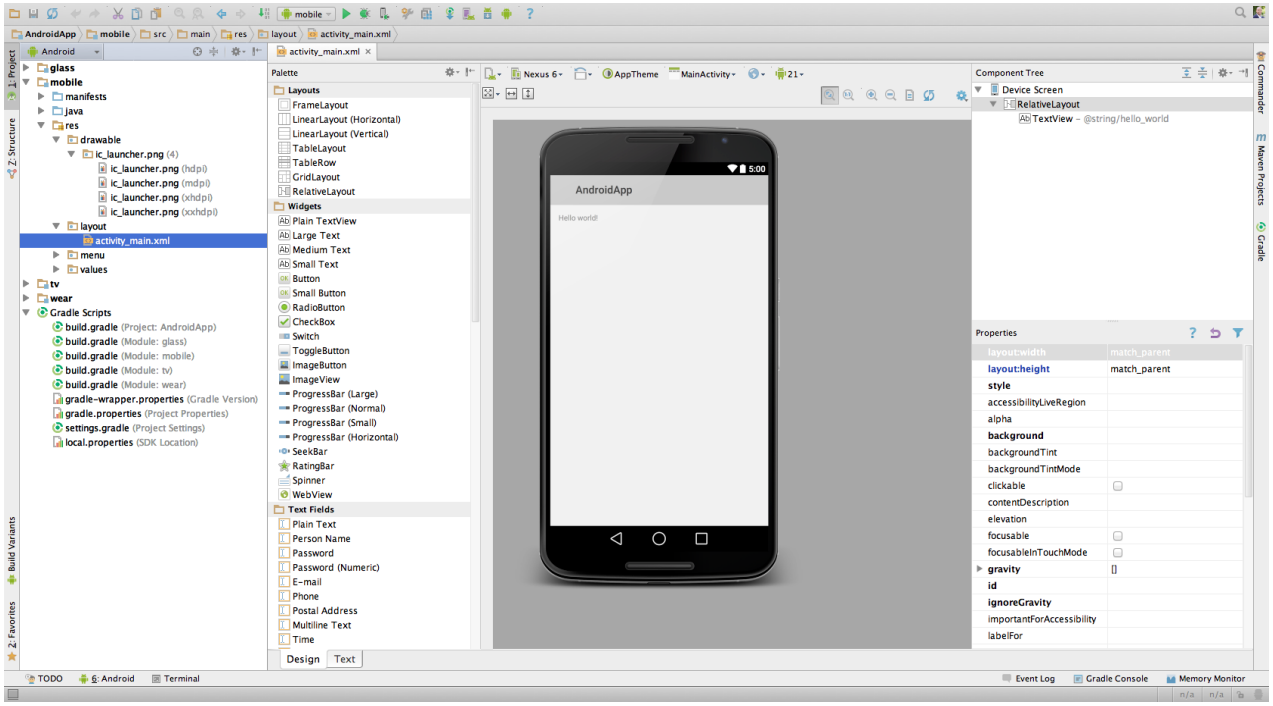


Figure 17: Android Studio Screenshot

Figure 17 shows the layout structure of the IDE. The left side of the IDE is Project Explore, next to it is the widget for components. In the middle is the screen for the development and the right side is dedicated to the properties. The IDE is customizable as required by the developer.

2.2.2 Xcode IDE

Xcode is an integrated development environment (IDE) containing a suite of software development tools developed by Apple for developing software for OS X and iOS. It is tightly integrated with the Cocoa and Cocoa Touch frameworks and provide many rich features. Figure 18 shows the layout structure of the IDE.

Source Editor: Write code using a professional editor with advanced code completion, code folding, syntax highlighting, and message bubbles that display warning, errors, and other context-sensitive information in line with the code in question.

Assistant Editor: The Assistant button splits the editor in two, creating a secondary pane that automatically displays files that are most helpful based on the code being actively edited. It can show the header counterpart, the superclass, callers, callees, or other helpful files.

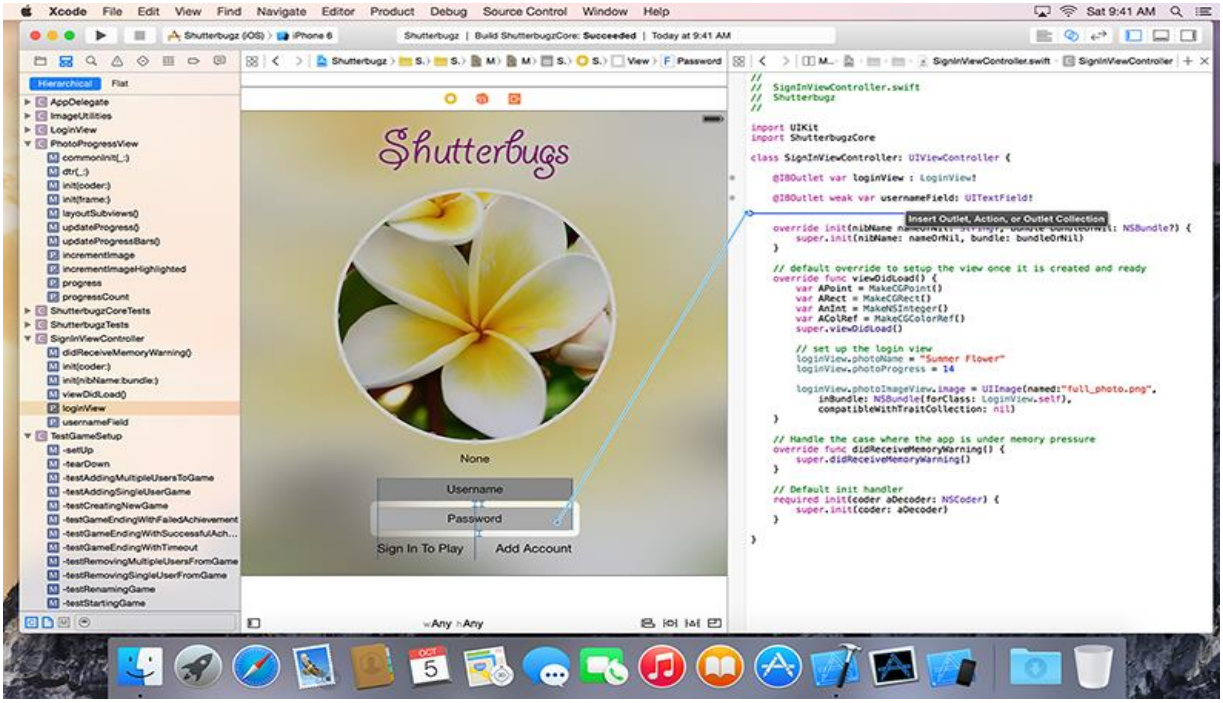


Figure 18: Xcode Screenshot

Version Editor: Xcode's Version editor displays a running timeline of commits, helps to determine blame, and graphically goes back in time to compare source files, with full support for Subversion and Git source control (SCM) systems.

Interface Builder: Built-in design and test the user interface without writing a line of code, prototype in minutes, then graphically connect the interface to the source within the Xcode editor.

iOS Simulator: iOS Simulator With the iOS SDK, Xcode can build, install, run, and debug Cocoa Touch apps in a Mac-based iOS Simulator for a streamlined development workflow.

Integrated Build System: Handles the most complex builds, scaling to maximize the power of multi-core Macs, and will automatically sign, provision, and install iPad and iPhone apps onto a device.

Compilers: The powerful open-source LLVM compiler for C, C++, and Objective-C is built into Xcode and available from Terminal. With it, the code compiles quickly, and is optimized by Apple to produce blazing-fast apps specifically tuned for the CPUs in iPhone, iPad, and Mac.

Graphical Debugger: Debugs the app directly within the Xcode editor. Hovers over any variable to drill into its contents, uses Quick Look to see the data it contains, or right-click to add the variable to the watch list.

Continuous Integration: Xcode Server, a feature of OS X Server, controls server-side bots that continuously build, analyze, test, and even archive the Xcode projects. The Xcode IDE configures these bots, analyzes nightly build and test results, and can track down which check-in broke the build.

This is just few a features of the Xcode IDE in brief and more can be explored as required for the development of the required application.

2.3 PCB Designing and Development

This section describes more in detail the PCB designing and development, the tools used for the PCB designing and for the simulation of the circuit. The core of the hardware is circuit board and its make or break stage for any prototype development.

2.3.1 KiCad

KiCad is open source software for electronic design automation (EDA) or electronic computer aided design (ECAD) with advanced features of Schematic Capture, PCB layout, Gerber file generation/visualization and library editing. It facilitates the design of schematics for electronic circuits and their conversion to PCBs (printed circuit board) design. The tools also include the packages for creating a bill of materials, artwork and Gerber files, and 3D views of the PCB and its components. Figure 19 shows the layout structure of the IDE. [24]

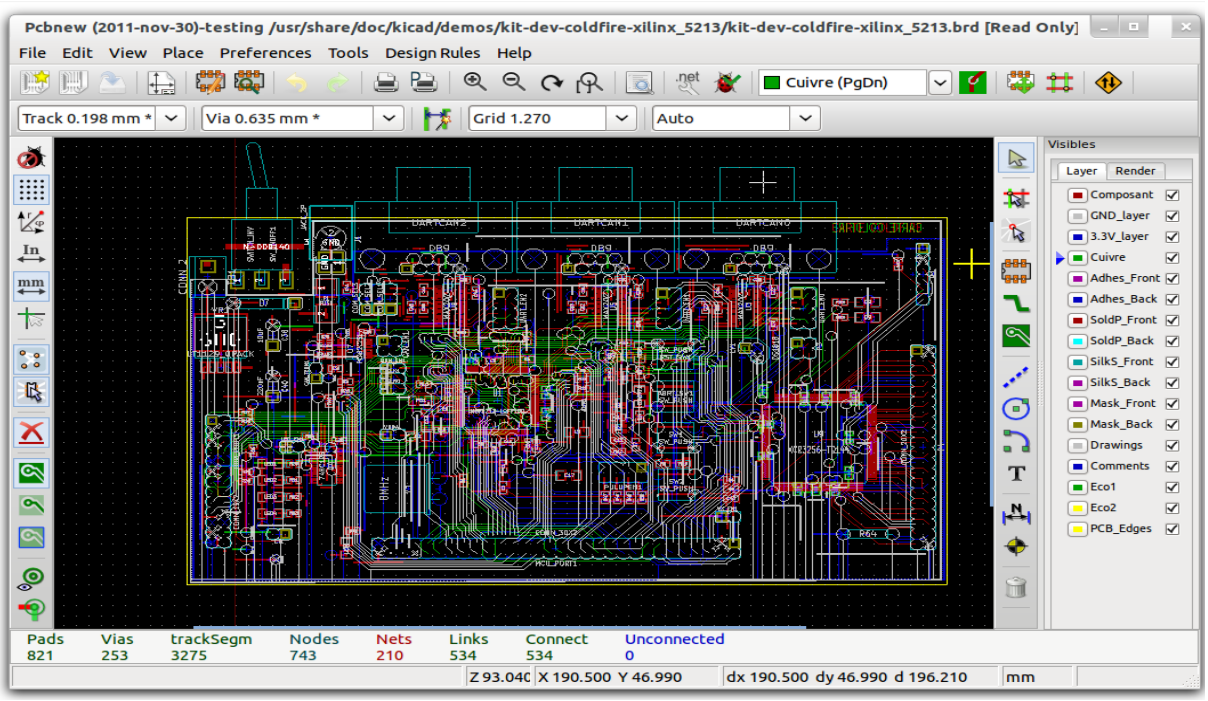


Figure 19: KiCad Screenshot

The KiCad suite has five main parts:

a. KiCad - the project manager

b. eeschema - the schematic capture editor

c. cvpcb - footprint selector for components used in the circuit design.

d. pcbnew - the PCB layout program. It also has 3D View.

e. gerbview - the Gerber (photo plotter documents) viewer.

2.3.2 LTspiceIV (Simulation tool)

LTspice IV is a high performance SPICE simulator, schematic capture and waveform viewer with enhancements and models for easing the simulation of switching regulators.

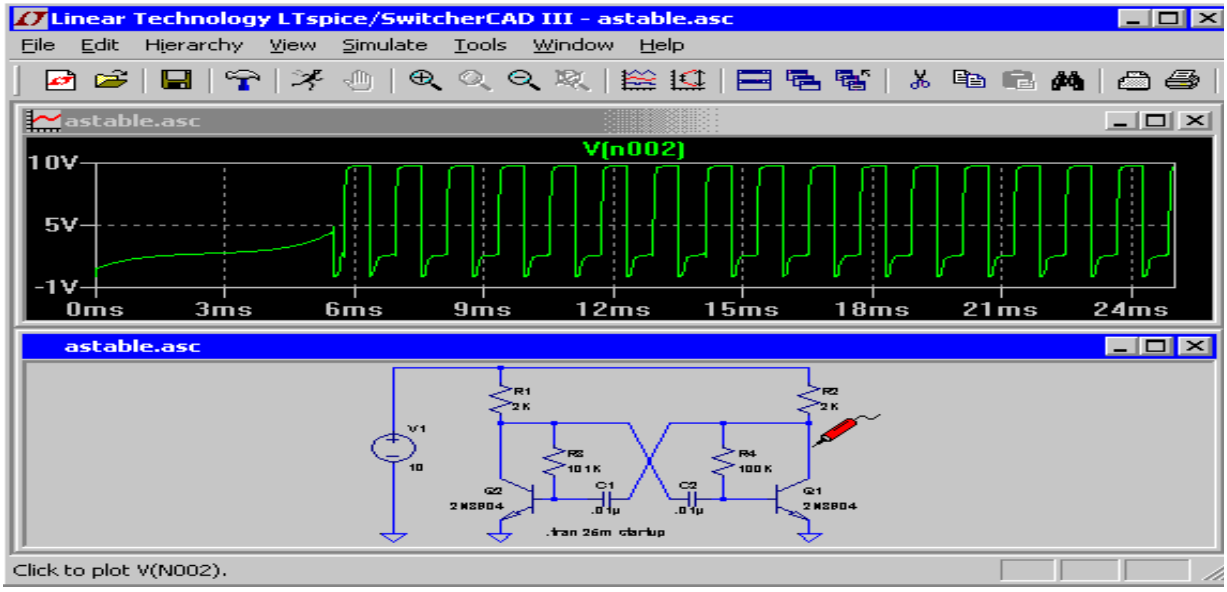


Figure 20: LTSpiceIV Screenshot

Basically SPICE (Simulation Program with Integrated Circuit Emphasis) is a general-purpose, open source analog electronic circuit simulator. It is a powerful program that is used in integrated circuit and board-level design to check the integrity of circuit designs and to predict circuit behavior. Figure 20 shows the layout structure of the IDE.

This chapter introduces the application development and low level development of the project. The next chapter focuses more on the Low Energy Technology and its functionalities, as well as the structure of the protocol.

3. Low Energy Technology

Bluetooth LE is a new technology, an alternative to classic Bluetooth, not a replacement. It is created to support completely different use cases and market segments than its parent. LE is intended to be used in devices which have to operate with a limited amount of energy (e.g. coin-cell battery) for very long periods of time. These devices can be any kinds of detectors, indicators, or sensors. It is not useful in products such as wireless headphones which involve long connection periods in which large amount of data is exchanged. It has been assumed to consume only a fraction of power compared to classic Bluetooth. Figure 21 illustrates more about Bluetooth Protocol Stack in different roles. [9]

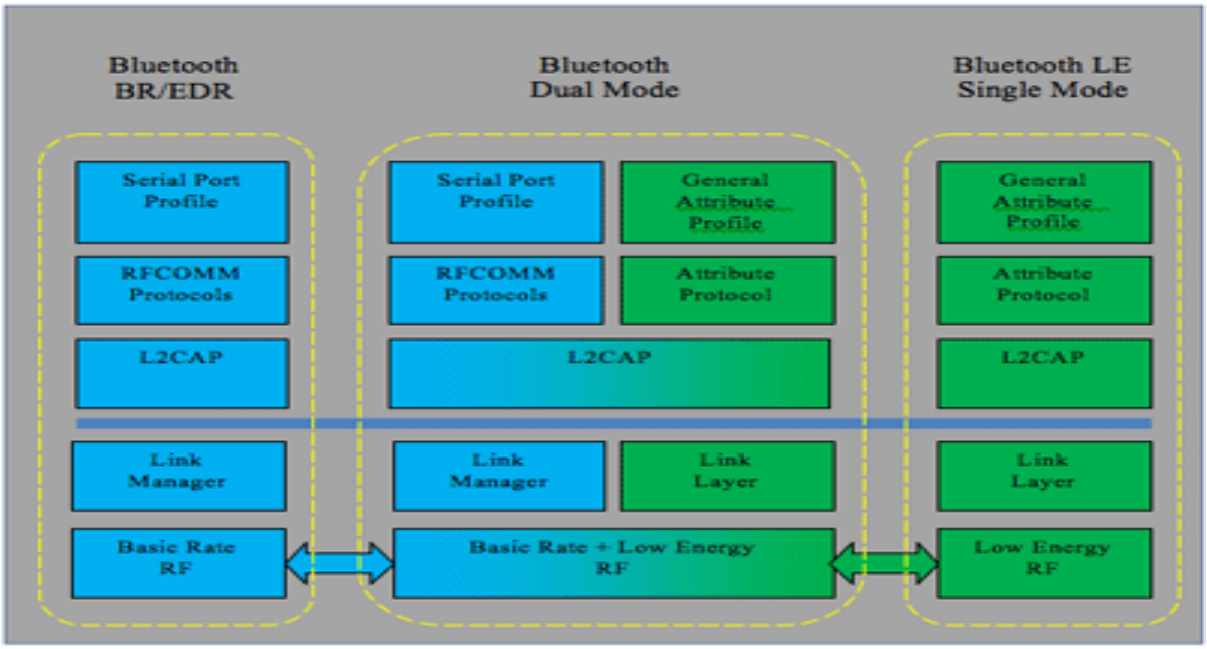


Figure 21: Bluetooth Protocol Stack

Bluetooth Smart and Smart Ready are new marketing names for Bluetooth enabled products. Smart is the name of the product that carries LE technology inside and Smart Ready is the name for a hybrid product that supports classic and LE technologies. Typical

smart products are peripheral devices which have a very limited energy capacity like car keys, TV-remote controllers, or any type of sensor units. Smart Ready products are central devices such as smart-phones, tablets, or TVs. [7]

3.1 Radio Technology

LE operates in 2.4 GHz unlicensed ISM radio band. The spectrum range is from 2402 - 2480 MHz, which is divided into 40 individual 2 MHz wide RF channels. The technology uses a technique called frequency hopping to combat against interference at the wildly used spectrum. Three of the 40 channels are used only for broadcasting advertisements and these are located in the spectrum as far from the known channels used by the Wi-Fi access point as possible to allow robust device discovery. [10, 11]

3.2 Functionality

LE technology will provide capabilities to advertise, scan, and connect. These capabilities can be used to discover other devices and to be discoverable to other devices as well as connect to other devices and be connectable to other devices. A device could use advertising to inform about its presence to nearby devices and even include small amounts of dynamic data to the advertisement, e.g. services that the device supports. A device that uses scanning functionality listens to these advertisements. After receiving one advertisement, it may attempt to request more information of the advertiser (i.e. a scan request) or initiate connection with it (i.e. a connection request). When the connection is established, the advertising device becomes a slave of the connection and the initiating device becomes the master. A master could connect with multiple slave devices at the same time, but a slave can be connected with only one master. Basically this means that

the master can be connected with one or more slave devices and at the same time scan for other devices. A slave could also advertise while connected to the master, but that advertising shall not lead to connection. [11]

3.3 Bluetooth Device Address

Every Bluetooth device has a unique device address that identifies that particular device from the others. The device address is a 48-bit MAC address that is divided into two. The first half of the address is company specific and the other half is device specific. There are also two types of addresses that the device may use. [11]

3.3.1 Public Device Address

A public device address is a fixed address that is assigned to a particular device. It must be registered with the IEEE Registration Authority and must never be changed during the lifetime of the device. [10, 11]

3.3.2 Random Device Address

A random device address shows to other devices just like the public address, but it is periodically changed and randomly generated. It hides the identity of the device from strangers but may be resolvable if desired (i.e. private address). [11]

3.4 Security

Technology has taken security considerations into account by supporting encrypted connections and private device addresses to combat against eavesdropping and man in the middle attacks. A procedure called pairing may be used to create temporary encryption

with any ongoing connection with the approval of both devices. A more permanent encryption procedure is called bonding, where both devices agree to share and save an encryption key (i.e. a shared secret) to be used to encrypt the link in future connections. A private device address is a random address that changes periodically, but is resolvable for the bonded device. This can be used to hide the identification of the device from third parties. [11]

3.5 Protocol Stack

The protocol stack shown in Figure 22 describes the software implementation of the protocols according to which Bluetooth LE operates. As can be seen in the Figure 22, it follows the same layered architecture as classic Bluetooth.

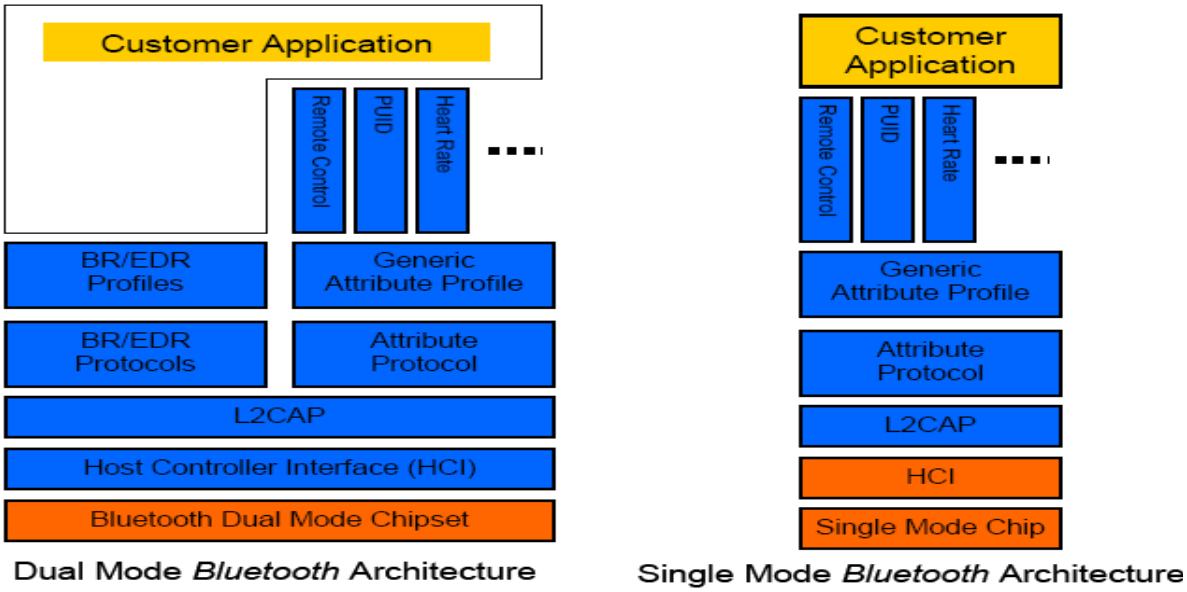


Figure 22: BLE Software

The figure also shows how the protocol stack is split up in two entities: controller and host. This division is made on the basis that both entities have very different timing requirements from the system they are running on. As a result, this allows them to be

implemented separately. The controller includes the hardware for the radio and the software that controls it. It is typically implemented separately in the form of a small chip, because the radio control has strict requirements for the timings. [7, 10, 11]

The host entity is a pure software stack that is responsible for higher level functioning providing an application programming interface (API) to the application and profiles. It has many more relaxed timing restrictions compared to the controller. That is why it is often implemented directly to the application processor along with the user's application. These two entities are connected through Host Controller Interface (HCI), which is delivered either logically (API) or physically (UART) depending on the implementation. The host is controlling the controller with HCI commands defined in the Bluetooth specification. Generally, the controller is often referred to as the Bluetooth hardware and the host is referred to as the Bluetooth stack. [7, 11]

3.6 Device Discovery

Advertising and scanning procedures are the foundation of every application that is designed around Bluetooth LE. These procedures used together enable device discovery. Without device discovery, LE devices cannot be aware of each other. Without awareness, the devices could not connect with each other and without connection they could not exchange data. A device using advertising is called an advertiser and a device that is using scanning is called a scanner. How these procedures work in simple terms is illustrated in Figure 23. [11]

The advertiser broadcasts itself to the nearby terrain and the scanner listens out for these advertisements. In order to utilize these procedures in an application, a more thorough explanation is well justified. The LE packet format is shown in Figure 23 for all data that is exchanged through air. The PDU field showing in the figure holds the actual payload inside.

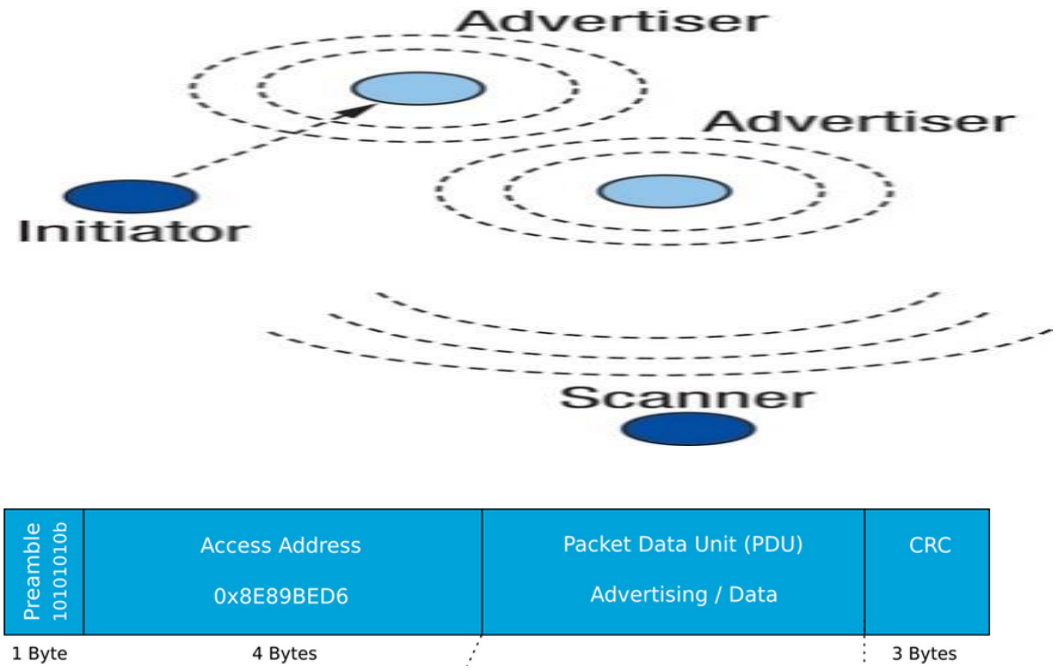


Figure 23: BLE Discovery and LE Packet format

LE defines two types of PDUs: advertising and data. The maximum packet is 376-bit long and the minimum packet is only 80-bit. The LE radio works with a 1 Mbps bit rate, which means that the longest possible packet would take 37 Second of all is the concept of channels.

As mentioned before, LE technology divides the over the air traffic into 40 RF-channels. The channels are indexed and spread over a 2.4 GHz ISM-band. The channels are divided into two groups: advertising and data. Channels 37, 38, and 39 are advertising channels and only used for device discovery purposes. They are located on the band as far away from each other as possible to enable robust device discovery even in bad conditions with the principle that at least one channel is accessible anytime. Another factor

which has affected the location of the advertising channels is the Wi-Fi access point, which uses the same ISM-band. [11]

After the devices have discovered each other, they may decide to form a more permanent relationship, a connection. A connection is initiated when the initiator device sends a connection request as a response to the advertising packet. An initiator is a device which acts just like a scanner but is only interested in the specific advertiser. When the connection is initiated, the initiator becomes the master and an advertiser becomes a slave of the connection. [11]

3.7 Limitations

The technology with the lowest possible power consumption has its limitations. The speed of data over the air from application to application, i.e. application throughput, has the absolute theoretical maximum of 250 kilobits per second with bulk transmission. However, when taking into account possible signal fading, interference, and packet overhead, a more realistic estimate for application throughput is about 5 kbps. Still, maintaining this kind of speed for longer periods of time would draw a relatively large amount of power and thus kill the concept of low power. Second of all, the coverage area (i.e. range) is typically not more than 10-15 meters. It is not restricted directly by the specification but by the power consumption. The more power is used to transmit the signal, the further it will radiate. It is left for the user to decide the best range power consumption ratio. [7, 11]

The next Chapter focuses on the prototype development process related to the project. It also elaborates the PCB designing in brief and discusses more details of software development.

4. Prototype and Software Development

Prototype development was initiated with the study of requirements and the requirement specifications first draft was prepared. According to the specification, hardware component was shortlisted. For the communication medium, BLE was the best match due to low power consumption and 10-15 m radius range. Datasheets of all components were searched and requested from the manufacturer if not otherwise available. The study was began with hardware designing.

4.1 PCB Designing and Assembling

This is the first step toward the hardware designing. A respecting schematic diagram with KiCad was drawn depending on the component requirement. Figure 24 gives a small section of the KiCad schematic for the new hardware designing process.

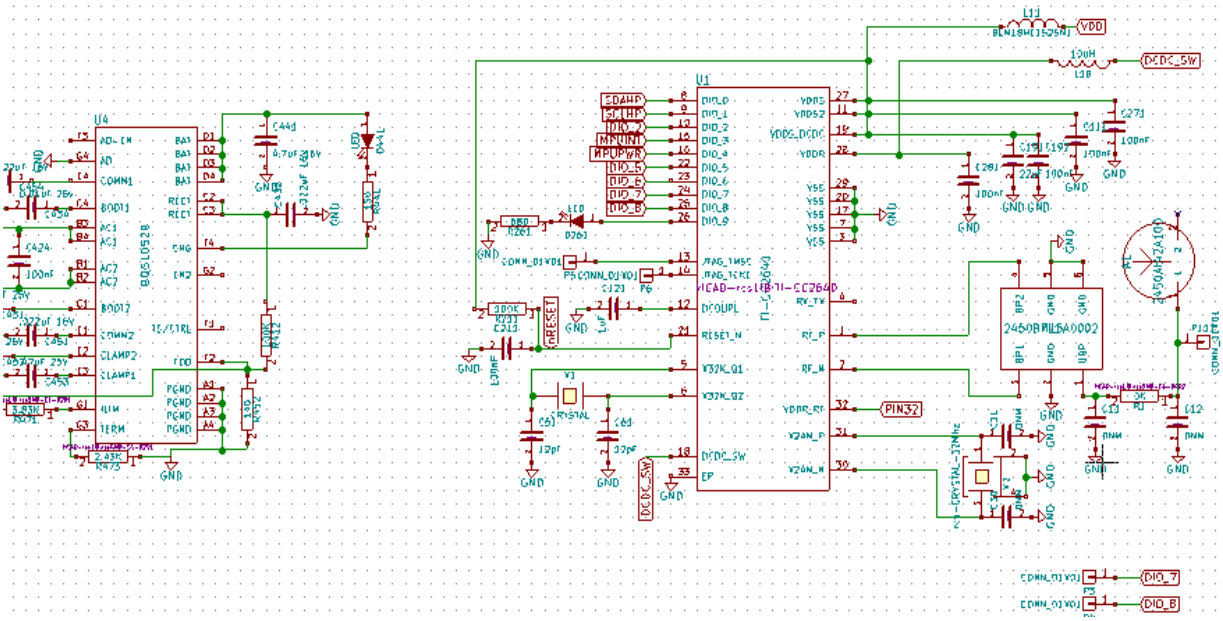


Figure 24: Schematic Diagram

Based on the schematics the PCB layout was plotted. It took a few days of work and rework on component placing and minimizing the space requirement. At the same time it was higher priority not to ignore the PCB designing basic requirements. KiCad was used for the PCB layout plotting. Figure 25 represents the PCB layout, Figure 26a is the top layer of the PCB and Figure 26b is the bottom layer.

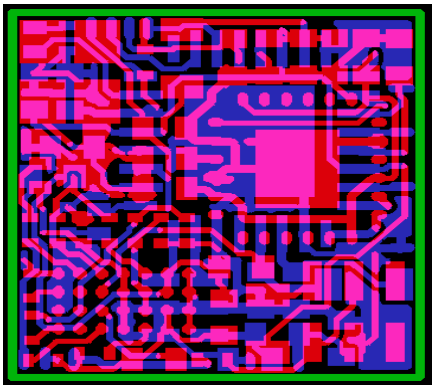


Figure 25: PBC Layout

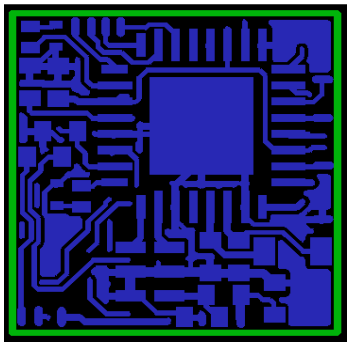


Figure 26a: Top Layer

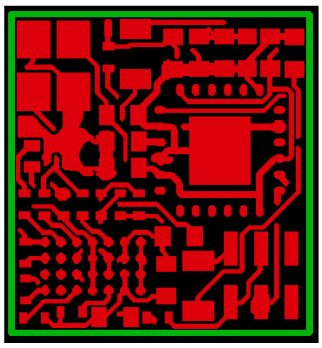


Figure 26b: Bottom Layer

PCB Assembling

After a few cycles of the assembling the final hardware came in shape. Figure 27 depicts the result.

Figure 27: PCB Board

The assembly of the components on the PCB was done by machine due to its size and complexity. This type of assembly is usually done in a factory at very a large scale.

4.2 Software Stack for New Hardware

The software stack to new prototype hardware of Figure 27 was developed from scratch. Now the software brings the newly developed prototype hardware into life and the required software was ported to the new PCB Board. In Figure 28 the relevant code is illustrated which made PCB board up and running. The figure also highlights some commands to read/write and dump collection.

```
root@ubuntu-vm:~# hciconfig hci0 leadv
LE set advertise enable on hci0 returned status 12
root@ubuntu-vm:~# hcitool -i hci0 cmd 0x08 0x0008 1e 02 01 1a 1a ff 4c 00 02
15 48 45 4c 4c 4f 57 4f 52 4c 44 00 00 00 00 c5 00
< HCI Command: ogf 0x08, ocf 0x0008, plen 26
 1E 02 01 1A 1A FF 4C 00 02 15 48 45 4C 4C 4F 57 4F 52 4C 44
 00 00 00 00 C5 00
> HCI Event: 0x0e plen 4
 01 08 20 00

root@ubuntu-vm:~# hcidump -RX
HCI sniffer - Bluetooth packet analyzer ver 2.5
device: hci0 snap_len: 1500 filter: 0xffffffff
< 0000: 01 0c 20 02 00 01                                        .. ...
> 0000: 04 0e 04 01 0c 20 00                                     ..... .
< 0000: 01 0b 20 07 01 10 00 10  00 00 00                        .. ........

 0010: 41 42 43 44 bb                                           ABCD.
> 0000: 04 3e 2a 02 01 00 00 15  25 69 70 f3 5c 1e 02 01  .>*.....%ip.\...
 0010: 1a 1a ff 4c 00 02 15 e2  c5 6d b5 df fb 48 d2 b0  ...L.....m...H..
 0020: 60 d0 f5 a7 10 96 e0 00  00 00 00 c5 b5           `............
> 0000: 04 3e 19 02 01 04 00 15  25 69 70 f3 5c 0d 0c 09  .>......%ip.\...
 0010: 75 62 75 6e 74 75 2d 76  6d 2d 30 b6              ubuntu-vm-0.
> 0000: 04 3e 0f 02 01 00 00 13  59 19 04 a5 78 03 02 01  .>......Y...x...
 0010: 05 ab
```

Figure 28: Code Snippet

The data dump from the prototype is also illustrated in Figure 28. This is a very early stage communication channel establishment between the developed smart device prototype and standalone PC running Linux OS.

4.2.1 GAP Role

GAP acts as the basis for all other Bluetooth profiles. The connection establishment and discovery between two devices are defined by GAP. There are four GAP roles which are central and peripheral, observer and broadcaster. Here the study only focuses on central and peripheral, since the broadcaster and observers never actually enter connection. For this project the Android device will act as central and the developed hardware as peripheral device. In practice this means that the smart device implements GATT server which stores data that the client (Android application) reads. Figure 29 show the role of GAP in all type of Bluetooth. [7]

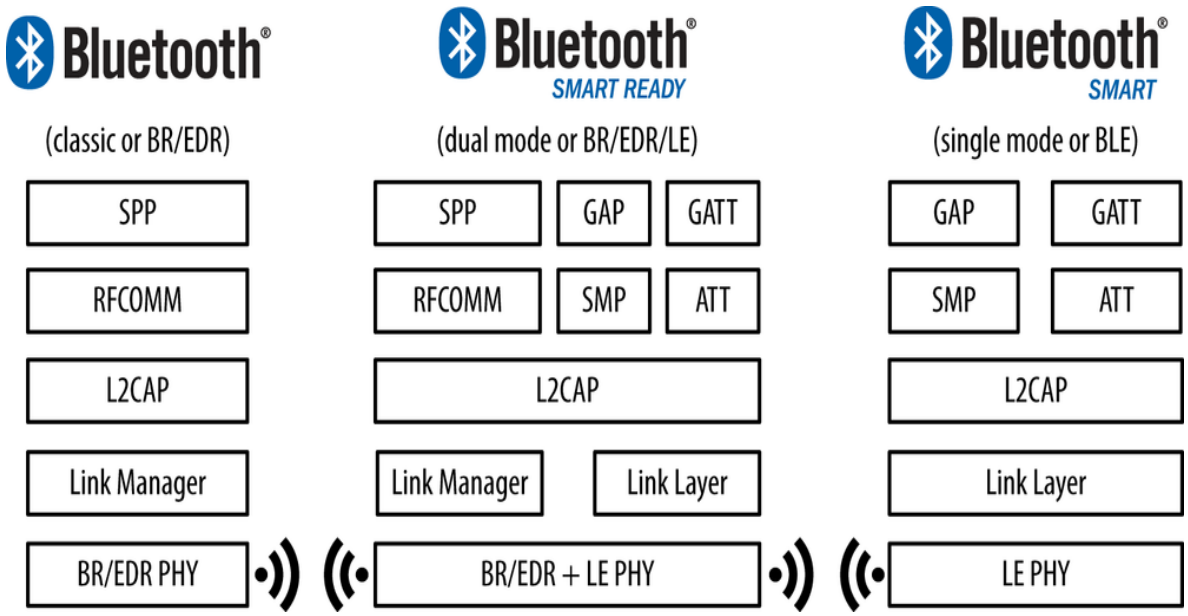


Figure 29: GAP in Bluetooth

Figure 29 above shows the three type of Bluetooth i.e. Bluetooth Classic, Bluetooth Smart Ready and Bluetooth Smart. The composition of the stack protocol among three clearly show the internal difference, but the lower layer and top layer have similar kind of protocol. As Bluetooth classic is old technology, it has clearly a different stack structure and usage compared to the other two. The wireless coverage range and pattern is common among them.

4.2.2 GATT (Generic Attribute)

GATT is an abbreviation of Generic Attribute profile. It is used by Bluetooth Smart devices. GATT specifies the transportation and storing operations and common framework for ATT. The data format on GATT server is also dictated by this. In low energy devices it is used for discovering services on device. The two roles of GATT are client and server. Figure 30 illustrates the GATT protocol.

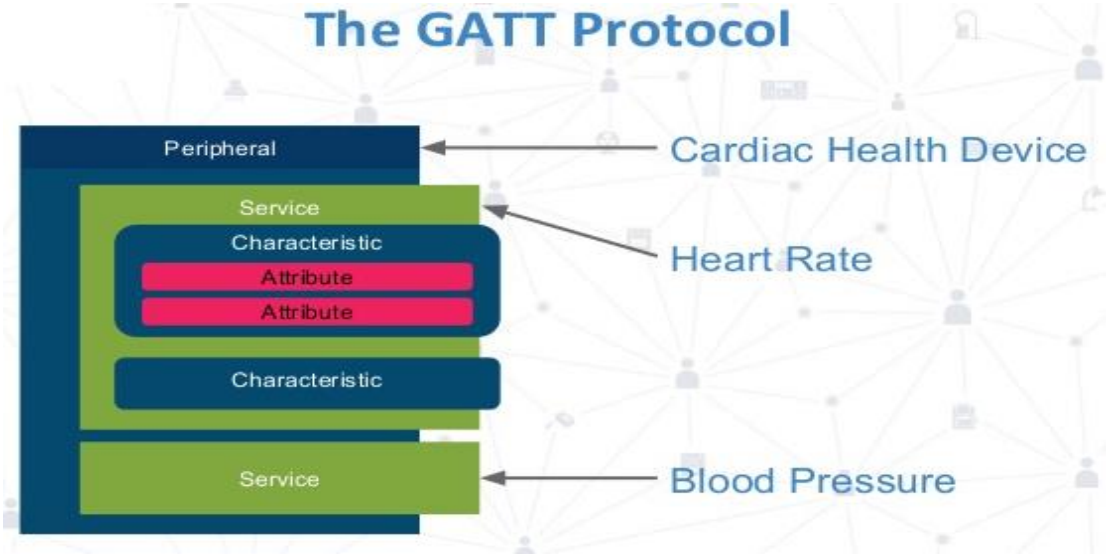


Figure 30: GATT Protocol

GATT client acts as a client which reads data from the server. The client is the device which uses the data for some application of Figure 30. The GATT client sends protocol requests to the server. The server accepts the requests and can send notifications and other information to client.

The server offers characteristics and services to client. Characteristics are values that are used in services. Together with value access and display information the value can be retrieved by client. Services are functions and accompanying data that are associated with different kind of behaviors. For example, in a smart device all different sensors are separate services and the characteristics are the values obtained from these

services. Figure 32 demonstrates the working functionality of the prototype and how to read and write characteristics with different values.

```
root@ubuntu-vm:~gatttool -i hci0 -b B4:99:4C:64:CD:F9 --characteristics
handle = 0x0002, char properties = 0x02, char value handle = 0x0003, uuid = 00002a00-0000-
1000-8000-00805f9b34fb
handle = 0x0004, char properties = 0x02, char value handle = 0x0005, uuid = 00002a01-0000-
1000-8000-00805f9b34fb


root@ubuntu-vm:~# gatttool -i hci0 -b B4:99:4C:64:CD:F9 --char-desc
handle = 0x0001, uuid = 2800
handle = 0x0002, uuid = 2803
handle = 0x0003, uuid = 2a00
root@ubuntu-vm:~# gatttool -b B4:99:4C:64:CD:F9 --interactive
[   ][B4:99:4C:64:CD:F9][LE]> connect
[CON][B4:99:4C:64:CD:F9][LE]> char-read-hnd 0x25
[CON][B4:99:4C:64:CD:F9][LE]>
Characteristic value/descriptor: 00 00 00 00
[CON][B4:99:4C:64:CD:F9][LE]> char-read-hnd 0x25

Notification handle = 0x0025 value: e9 fd 84 0e

root@ubuntu-vm:~# gatttool -b B4:99:4C:64:CD:F9 --interactive
[   ][B4:99:4C:64:CD:F9][LE]> connect
[CON][B4:99:4C:64:CD:F9][LE]> char-read-hnd 0x25
[CON][B4:99:4C:64:CD:F9][LE]>
Characteristic value/descriptor: 00 00 00 00
[CON][B4:99:4C:64:CD:F9][LE]> char-read-hnd 0x25
[CON][B4:99:4C:64:CD:F9][LE]>
Characteristic value/descriptor: 00 00 00 00
[CON][B4:99:4C:64:CD:F9][LE]> char-write-cmd 0x29 01
[CON][B4:99:4C:64:CD:F9][LE]> char-read-hnd 0x25
[CON][B4:99:4C:64:CD:F9][LE]>
Characteristic value/descriptor: 1e fe d0 0d
[CON][B4:99:4C:64:CD:F9][LE]> char-read-hnd 0x25
[CON][B4:99:4C:64:CD:F9][LE]>
Characteristic value/descriptor: 9e fe 3c 0e
[CON][B4:99:4C:64:CD:F9][LE]> char-write-cmd 0x26 0100
[CON][B4:99:4C:64:CD:F9][LE]>
Notification handle = 0x0025 value: e9 fd 84 0e
[CON][B4:99:4C:64:CD:F9][LE]>
[CON][B4:99:4C:64:CD:F9][LE]> char-write-req 0x0067 81
[CON][B4:99:4C:64:CD:F9][LE]> Characteristic value was written successfully
char-write-req 0x0067 82
[CON][B4:99:4C:64:CD:F9][LE]> Characteristic value was written successfully
char-write-req 0x0067 00
[CON][B4:99:4C:64:CD:F9][LE]> Characteristic value was written successfully
char-write-req 0x0067 82
[CON][B4:99:4C:64:CD:F9][LE]> Characteristic value was written successfully
char-write-req 0x0067 81
[CON][B4:99:4C:64:CD:F9][LE]> Characteristic value was written successfully
char-write-req 0x0067 00
[CON][B4:99:4C:64:CD:F9][LE]> Characteristic value was written successfully
```

Figure 31: Code Snippet

5. Prototype Integration and Testing

This section goes through the design and implementation of layout with the help of the XML programming language and Android Studio IDE. Java classes and methods to meet the program requirements are also introduced.

5.1 Application Development

This section is more focused on the application development for smartphones which sync the data from smart device prototype. It also describes more in detail the Android Studio IDE and project structure.

5.1.1 Reviewing Requirements

With respect to smart device hardware there are several needs for the application prototype. During this process a prototype which hosts only part of the stated requirements was created. The application connection to smart device over BLE was the focus area.

5.1.2 Structure of Android project

The structure of the Android project is mostly the same, but also may differ depending on the project needs and IDE tool. The basic structure is explained while using the Android Studio. When a programmer uses the Android Studio IDE, the project structure is generated automatically with the sample “Hello World” application. The GUI of Studio is very friendly and convenient to use.

Project folder structure: A basic Android project would have six directories: *assets, bin, gen, libs, res, src*. Also there are some files in the project root directory such as:

AndroidManifest.xml, licenses, project properties and other files. The most important for the developers are “res” and “src” directories. The “res” directory contains all the current project resources such as: images, layouts, custom strings and other values. Images are stored in different directories depending on their size, so that the application can automatically choose the right image depending on the device specifications. Layouts are stored in the “layout” folder. Basically a layout file example would be an XML file which would specify elements and their position in the current view. Also it is possible to code custom strings and colors so the parser can display them in the application. It is a recommended approach to store them in the values directory rather than hard code to the actual code or XML file. It would make further development easy at translating the application to other languages. The other important directory is “src”. This directory usually consists of Java files which are adding functionality to the application. The developer has options to create classes as separate Java files. If the class is created in GUI, the tool would generate automatically the statement in an “*Android Manifest*” file. If another programming environment is used, the user must specify any new class activity by hard coding the “*Android Manifest*” file. The Android manifest file usually is placed in the root directory of the project, and state required version of Android, needed permissions and all activities which are run within the application. [13]

Designing of the layouts: With reference to the requirements, four different views needs to be designed. Figure 32 shows the layout structure. The first is “Home”, the second is Connected, the third is Account and the fourth is Setting layout.

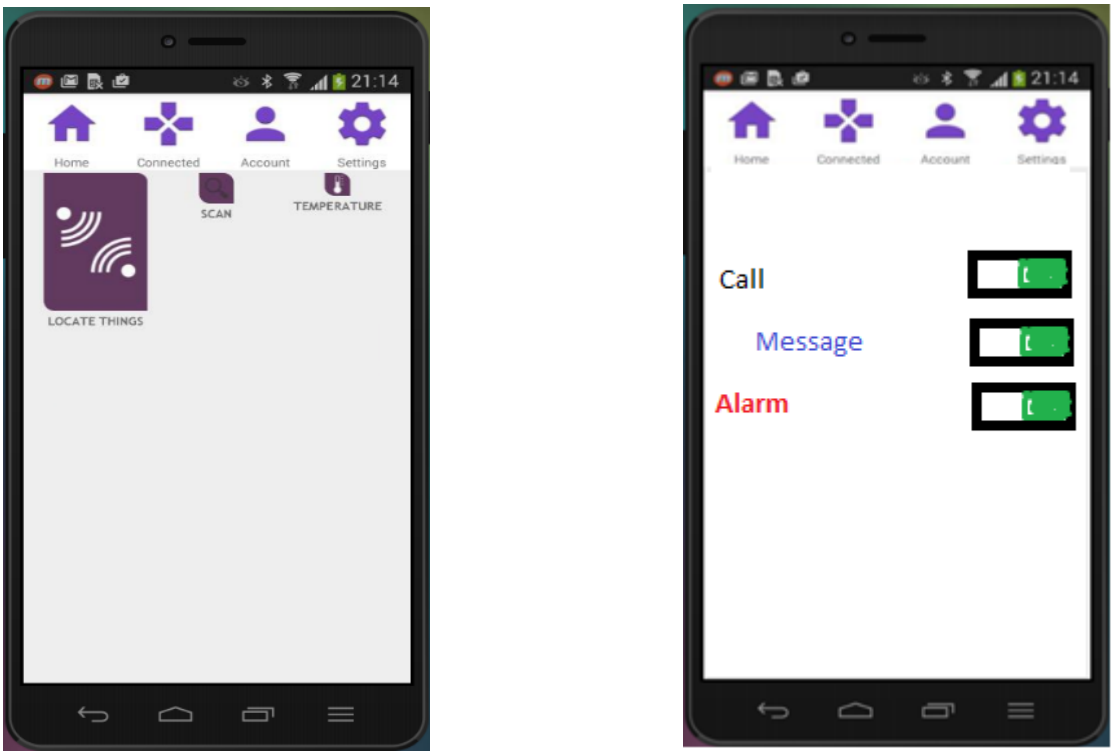


Figure 32: SmartSOS Home Screen and Settings Template.

Implementing the layouts: Android application uses XML layout for displaying its content. The XML document would consist of several tags with given properties. The parent tag would state the type of view which is the main for the document. Also it is allowed to use several views inside of one main view. Like any other XML tag, this tag is given several properties which will define its identification, style, on-click action, etc. Identification is one of the important part of the tag.

By defining the “id” the programmer can use it in the Java code. Styles can be hard coded in the statement or linked to a separate file which will specify the style for this element. Action properties can call the event action which is defined in the code part of the project. The main view of the prototype application will have a simple layout. [14]

For organizing as application design the LinearLayout statement is used and it is also the parent layout statement for main view XML document. That is why it has to be linked to the http://schemas.android.com/apk/res/android. As can be seen, this statement

hosts some properties which are important for displaying of this document. xmlns:android could define the namespace information, but in this case it just defines the unique parent tag. Layout_width tells to the mobile device how to display this view. In this case the option “fill_parent” is used so the program will use all the space on horizontal scale. Layout_height is the same as width but it uses vertical scale to display the view. Orientation is the display view depending on the device orientation, it can be vertical or horizontal. In this case it is horizontal so it means that the device's screen is used in portrait mode. All the child entries will be entered between the end of definition and closing statement </LinearLayout>. The example of parent statement for XML document would look like this.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
</LinearLayout>
```

In the example code the type of layout is specified with standard Android XML schema. It also specifies the height, width and orientation characteristic. The parameters are mandatory for the implementation of LineaLayout in Android application. Figure 34 shows the code snippet of the manifest file from Android application. The manifest file holds all the characteristics of the application e.g. permissions, activities and their dependency. The code snippet gives a more organizes structure of the application and its basic permissions and usage. In any Android application the manifest file has a central role and everything is dependent on this. It is impossible to launch any application on Android platform without the manifest file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bluetoothgatt"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-feature android:name="android.hardware.bluetooth_le"
android:required="true"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity android:name=".MainActivity"
            android:label="smartSOS "
            android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity android:name=".BeaconKitKatActivity"
            android:label="Beacon Monitor"
            android:screenOrientation="landscape"
            android:enabled="@bool/use_kitkat">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".BeaconLollipopActivity"
            android:label="Beacon Monitor"
            android:screenOrientation="landscape"
            android:enabled="@bool/use_lollipop">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Figure 34: Manifest code snippet

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
android:layout_height="match_parent"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:stretchColumns="*">
    <TextView
        android:padding="@dimen/activity_horizontal_margin"
        android:gravity="center_horizontal"       android:textSize="42sp"
        android:text="Android Weather Station"
android:layout_height="64dp" />
    <TableRow>
        <TextView
            android:id="@+id/text_temperature"
            android:layout_column="0"
            android:textSize="@dimen/measurement_text_size"
            android:gravity="center_horizontal"/>
        <TextView android:layout_column="2"
            android:id="@+id/text_pressure"
            android:textSize="@dimen/measurement_text_size"
            android:gravity="center_horizontal"/>
    </TableRow>
    <TableRow>
        <TextView   android:layout_column="0"
            android:textSize="@dimen/label_text_size"
            android:gravity="center_horizontal"
            android:text="Ambient Temperature"/>
        <TextView
            android:layout_column="2"
            android:textSize="@dimen/label_text_size"
            android:gravity="center_horizontal"
            android:text="Barometric Pressure"/>
    </TableRow>
    <TableRow>
        <TextView            android:id="@+id/text_humidity"
            android:layout_column="1"
            android:textSize="@dimen/measurement_text_size"
            android:gravity="center_horizontal"/>
    </TableRow>
    <TableRow>
        <TextView            android:layout_column="1"
            android:textSize="@dimen/label_text_size"
            android:gravity="center_horizontal"
android:text="Humidity"/>
    </TableRow>
</TableLayout>
```

Figure 35: main_activity code snippet

Figure 35 shows the xml format code snippet of the main_activity from the Android application development. The code also highlights the structure of the application. The activities are the structure of the application. Activities are interlinked and come to focus as their role is defined. Typically at least one activity exists in every Android Application and depending on the complexity and structure of the application it may have more than one.

5.2 Application Integration

Integration of an application is the most challenging part of the project development. This section mainly focuses on mobile application and the requirements of the application. It also describes the structure of the application composition and basic terminologies used in the Android application development process. These are standard practices seen during the Android application development.

5.2.1 Intents

In Android development Intents are objects that allow an application to request another app component to take action. Intents can be used to start a service or to deliver broadcast or start an activity. Switching from one activity user interface to another can be done through Intents. Intents can be of two different types, explicit and implicit. Explicit intents are commonly used by an application to start another component in itself. The component is called with its specific name. Immediately after creation, the intent starts the component determined in intent. However, explicit intent is difficult to use to request action from an external application since the name of the component is not known. For these cases there are implicit intents. Unlike explicit intents, implicit intents are handled by

Android system which looks for the appropriate component in other applications to take action. This method eliminates the need to know the exact component name and use general action instead. Implicit intents make use of intent filters which are defined in the manifest file and describe what kind of intents the application takes care of. The system goes through manifest files of applications in search for appropriate intent filter. It is also possible that an Intent matches two or more components. In this situation the system prompts the user to choose which application to use.

5.2.2 Broadcast Receivers and Broadcast Intents

Broadcast intents are used to communicate between different application components. In addition, the Android system uses them to notify applications about system events. For example connected devices and chargers cause a broadcast intent to be sent. Via broadcast intents applications can also send messages to each other. A broadcast receiver, as the name suggests, is a component that listens to specific broadcast intents. A broadcast receiver allows a piece of code to be executed on demand. Broadcast receivers must be registered for them to start listening. A good practice is also to unregister receiver when it is no longer required.

5.2.3 Handlers

Android runs its application on one single thread, the user interface, main thread. This, however, seizes the execution of user interface code in cases where there is a lengthy process to be executed. Therefore these processes are taken care of in the background thread. Many components also run their processes in the background. This allows the user interface to work while there is on-going action in the background. The downside of background processing is that the user interface is not updated when the

process is run on different thread which are not allowed to communicate with the main thread. The handler makes it possible for background thread to send messages and runnable objects to the main thread and in this way communicate with it. The handlers are attached to the thread on which they are created. Then the handler can be used to post to the original thread from a different thread. This is commonly used to update user interface after a process in the background is complete. Another way to use the handler is to schedule tasks to be executed after a defined period of time.

5.3 Smart Device Initial Setup

Setting up the smart device sensor consists of several steps. This is the part where most of the work had to be done in terms of seeking information and studying the implementation. Guides and tutorials for the configuring sensor were almost nonexistent and therefore a large amount of work had to be done studying the source code of the sample program and sensor datasheet.

5.3.1 Layout

With respect to Section 5.1 the user interface was developed and the core functionality was the main focus area. The user interface was not fully implemented, but provided the structure for the final application development and the user interface layout with the background functionality. It only implemented few buttons, for starting and stopping the Bluetooth device scan, and a List View in which the found devices would be listed by their MAC code. The List View was made clickable for selecting a device. Clicking an item on the list also started enabling a connection to the selected device. [14]

5.3.2 Bluetooth Connection

Since Android 4.3 the Bluetooth low energy has been supported and there are specific methods for establishing this kind of connection. [10, 15] The connection establishing has 3 steps.

1) Android API has a class that includes Bluetooth adapter through which basic Bluetooth tasks are done. First Bluetooth is enabled on a device in case it is disabled. Turning Bluetooth on is done by calling Intent on Bluetooth adapter that prompts the user to turn Bluetooth on from Settings. Intent starts an external application which takes care of the rest.

2) Finding low energy devices with Bluetooth adapters LeScan method. The method then scans for low energy devices and returns the found devices. Adding the items on the list is done in the by creating a callback that does the adding after devices are found. This creates a connection with the device, however not much can be done with that connection.

3) Connecting to the GATT server is the final phase of enabling connection. This also the phase that enables sensor to be used and gather data from it.

5.3.3 Gathering data – GATT server callback

Turning the sensors on and gathering data from sensors are done with callback. [10, 18] The steps for turning the sensor on a gathering data are listed below.

1) First check if the GATT server is connected. After that discover services on the device. In this case all the different sensors on the smart device are individual services. The service discovery must be done before the sensors can be turned on.

2) After discovery the sensor is turned on. This is done by writing a code in byte format to characteristics configuration data space on smart devices.

3) After the sensor is turned on the initial value from the sensor is read from data space on the smart device. This way it is also made sure that the sensor is turned on and gathering data.

4) To get data continuously the smart device has a notification function, which sends data on specified time periods and allows application to be updated. In this step the notification function is turned on by writing a description to the data space of a selected sensor.

5) Now the device picks up data and the smart device notifies the application on regular intervals. However, the data from the sensor is raw data and has to be handled before it is on a human understandable form. A special class is used to turn data on human readable form.

6) Finally data is human readable and can be written out. The data are set on TextView box on the user interface of the application.

5.4 Testing

Android Mobile application testing can usually be done inside the development tools using the emulators and further tested using hardware devices. However, when it comes to Bluetooth applications it is required to carry out the development and testing on the specific hardware model under target and it is a must to have the Android device. In this project, the black box testing is used to carry out the mobile application testing for the development work. Black Box testing is described as a method of software testing that is focused on functionalities of an application. It is defined as a software testing technique whereby the internal structures (workings) of the item under test are not known to the tester. Along with development small relevant unit testing has been done. Testing details are not covered under this study. For black box testing some automated tests are run for

every code change by the continuous integration system (CIS). Continuous test execution ensures that errors are detected as early as possible. All automated tests are executed several times every day. Continuous integration system does not allow delivering the software package to a server before all automated tests pass. This practice encourages to fix detected bug as soon as it was discovered. Manual user tests are executed periodically as the features are developed. At the end of the project development phase all automated unit and end-to-end tests were passing.

In this chapter, prototype integration was described with processes involved in it. It also focused on application integration and testing. The full coverage of the Android application and designing with Bluetooth is crux of this chapter. The next chapter will be focus on shape and tools used for final product prototype. Also, it will cover the final stage of this project development.

6. Prototype as Product

3D modelling tools were used to make a small product prototype in the form of wearable ring. The respective developed prototype hardware was considered and designed to fit in the wearable ring. Blender was used for 3D modeling and for drawing 3D shape. Also, 3D printer was used to print the shape or output of the 3D modeling.

6.1 Using Blender

Blender is a free and open source 3D animation suite. Blender can be used to create 3D visualizations such as still images, video and real-time interactive video games. It is cross platform and runs equally well on Linux, Windows and Macintosh computers with a small memory and disk footprint. Its interface uses OpenGL to provide a consistent experience across all supported hardware and platforms. [6]

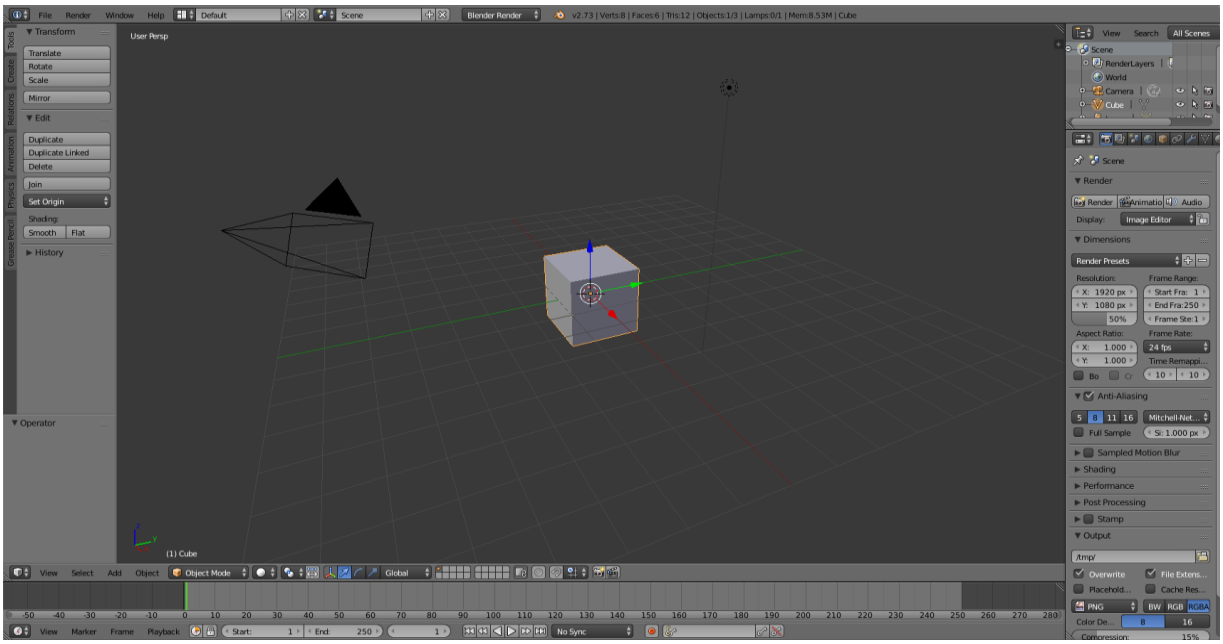


Figure 36: Blender Screenshot

Figure 36 illustrates the GUI of Blander. The square box in the middle of the GUI is default startup screen. On the left side of the Blander, object characteristics are listed

where on the right side of the window properties such as dimensions, location color etc. are listed. Blander provide very good documentation for new and advance users.

There are three UI principles which must be followed:

a. Non Overlapping: The UI is designed to allow one to view all relevant options and tools at a glance without pushing or dragging editors around.

b. Non Blocking: Tools and interface options do not block the user from any other parts of Blender. Blender typically does not use pop-up boxes (requiring users to fill in data before running an operation).

c. Non Modal Tools: Tools can be accessed efficiently without taking time to select between different tools. Many tools use consistent and predictable, mouse and keyboard actions for interaction.

The tools provide very advance playground to user. Many cycle and redesign may be required to reach desired shape, it is only matter of practice and remembering the tools to be used at right place.

6.1.1 Using 3D Printer

Ultimaker 3D printer was used to print the 3D modeling shaped ring from the blender. [23] Ultimaker 3D printer is a very good tool for prototype development and the result is unexpectedly fast as compared to past development tools. There are many pre-requisites before the usage of the Ultimaker and it is necessary to install the software to calibrate the machine. The preferred software for Ultimaker Original is Cura developed at Ultimaker. This software package prepares the 3D model into instructions that Ultimaker uses to produce an object. The main steps required are:

1. Most 3D printable files are in the STL format. This can be done with the help of Cura 2.

2. When the file is loaded into Cura, the progress bar is shown. Cura will automatically start making the 3D model.

3. In the meantime the settings and select a print type must be adjusted.

4. When the 3D model is prepared, the Save toolpath button pops up, and it gives the option to save the prepared model in a directory.

5. Now the included SD card is inserted into the computer, Cura changes the Save toolpath Button into the Toolpath to SD Button.

6. The Toolpath to SD button is chosen and the progress bar will run again, but now it will save the file on the SD card.

7. When Cura has finished saving the file, it will notify the success.

8. Before taking out the SD-card make sure the safely eject button in Cura.

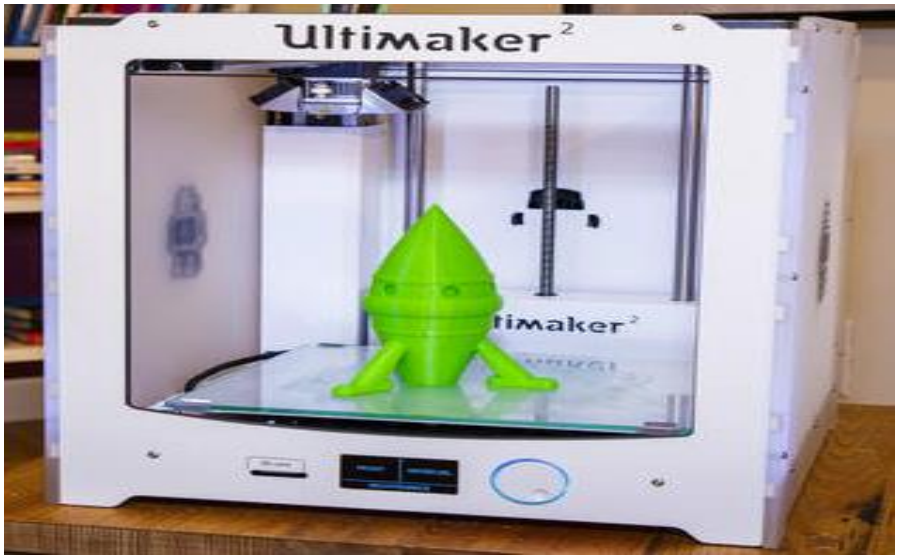


Figure 37: Ultimaker 3D Printer

Having done all this, the SD card is inserted into the Ultimaker 2 machine and the print button is pressed. After a couple of hours the final prototype was ready. Figure 37

illustrates the Ultimake 2 3D printer. Security measures are highly recommended as the material is very hot during the printing.

6.2 Final Shape of Smart Device

The final shape of the wearable smart device which was printed in the form of a ring with a 3D printer is demonstrated in Figure 38. The developed hardware prototype can be easily identified. This is a realistic picture of the product prototype.

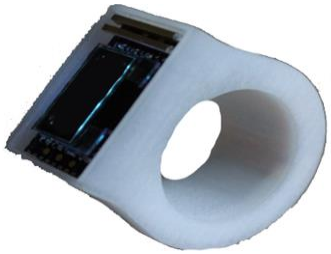


Figure 38: 3D printed ring

The size of the ring is very small and has limited space. With limited space and placing the developed hardware prototype into the ring was quite challenging. After some permutation and combination the prototype hardware was successful fitted and installed.

More testing and fine tuning was required before the first successful demo. Due to the limited scope of the thesis, further development is not covered under this study. The development of the prototype was the main focus of the thesis and it was achieved.

6.3 Future Development

There are many use cases of the developed prototype. A more finished product can be produced from this prototype with respective changes in the Android Application and BLE software stack. Figure 39 shows an imaginary usage of the prototype as a ring but the list can be long.

Figure 39: Imaginary usage

Many more features can be added as per the requirements such as motion movement or heart beat rate monitor. This prototype and combination of software for mobile application make the usage possibilities unlimited, the only requirement is to fine tune and fit into the final product shape as needed.

6.4 Summary

This chapter mainly focused on the final product development. The methods and technology used for product development were introduced and the end of the chapter introduces the final wearable smart device. The following chapter is the final conclusion of the study. Chapter 7 provides a crush of the study and project development.

7. Conclusions

The purpose of the study was to develop a prototype and a supporting application for a smart personal digital assistant. The advancement in technology and fast growing solutions make life easier, with less investment in hardware. The smart devices and sync applications on smartphones keep updating all the information from anywhere. They also provide the capabilities to prioritize the steps or actions. The technology and relevant applications in health care provide better solutions for a healthy life style. They also reduce the time and distance for the availability of medical service in an emergency.

7.1 Concept

The basic concept was to develop a wearable device which is always in sync with an application on a smartphone or on cloud via smartphone. The wearable device should be small in size with a very low power consumption. The user can adapt this smart device in his lifestyle without any changes. The functionality of device is very simple and based on a sequence of events. The user setups the application preferences and sync the smart device as per his requirement e.g. national emergency phone number and what needs to be done if something goes wrong. The smart device will be equipped with multiple inputs such as button or gesture recognition. In an event of input from the smart device, respective course of action will take place in a smartphone application such as calling an ambulance or making a phone call to the police.

7.2 Prototype Hardware/Software Selection

To meet the objective, a research was conducted on a new set of hardware and suitable open source development platforms were selected for the supporting software stack development of the prototype. The project thesis also covers the smart PDA application development for smartphone on Android platform. The project achieved the objectives set at the beginning.

7.3 Challenges and Future Development

The Master's thesis was for the author an opportunity to extend and discover new skills in Mobile Application Development with Smart Device. The goal set for the study was achieved. However, the final product is not 100% complete as the design of the user interface is customizable depending on the real requirement and implementation. The biggest challenge turned out to be the amount of time required for searching for information about sensors and their usage. In retrospective the biggest obstacles were the ones that taught the most and improved the author's application design skills.

However, due to limited time and scope the field testing of the smart PDA was not realized. The appropriate algorithm for initiating the emergency call and using the smart device whenever required should be adapted for future work.

References

[1] Fox, Susannah; Maeve, Duggan; Mobile Health 2012, 8.11.2012, Pew Internet & American Life Project;

URL:

http://www.pewinternet.org/~/media//Files/Reports/2012/PIP_MobileHealth2012_FINAL.pdf

Accessed on 10 May 2015

[2] Michael Essany, Mobile Health Care Apps Growing Fast in Number; mHealthWatch;

URL: http://mhealthwatch.com/mobile-health-care-apps-growing-fast-in-number-20052/

Accessed on 05 May 2015

[3] Prototyping Concept and Process

URL: http://www.innovate-design.com/prototyping-process/

Accessed on 10 May 2015.

[4] IoT Structure

http://www.gsma.com/connectedliving/wp-content/uploads/2014/08/cl_iot_wp_07_14.pdf

Accessed on 10 May 2015.

[5] IoT Usage

http://www.freescale.com/files/32bit/doc/white_paper/INTOTHNGSWP.pdf

Accessed on 01 May 2015.

[6] Blender Developer development guide

URL: http://www.blender.org/manual/getting_started/index.html

Accessed on 10 May 2015

[7] Bluetooth Special Interest Group. Bluetooth wireless technology [online]. Kirkland, WA, USA: Bluetooth Special Interest Group.

URL: https://www.bluetooth.org/Building/overview.htm

Accessed on 24 April 2015

[8] Bluetooth SIG. About Bluetooth SIG [online]: Bluetooth SIG, 2014.

URL:http://www.bluetooth.com/Pages/about-bluetooth-sig.aspx

Accessed on 02 May 2015.

[9] Galeev M. Bluetooth 4.0: An introduction to Bluetooth low energy – Part II [online]: EE Times, 28.07.2011.

URL:http://www.eetimes.com/document.asp?doc_id=1278966

Accessed on 28 April 2015

[10] Haydon R. Bluetooth low energy: The developer's handbook. Practice Hall; 2012.

[11] Decuir J. Bluetooth 4.0: Low energy [online]. Cambridge, UK: Cambridge SiliconRadio SR plc; 2010

URL: http://chapters.comsoc.org/vancouver/BTLER3.pdf

Accessed on 03 May 2015.

[12] Bluegiga Technologies. Bluetooth low energy [online]. Espoo, Finland: Bluegiga Technologies

URL: https://www.bluegiga.com/en-US/development-kits/

Accessed on 15 May 2015

[13] App Manifest, Android Developer's Guide,

URL: http://developer.android.com/guide/topics/manifest/manifest-intro.html

Accessed on 10 May 2015.

[14] Layout, Android Developer's Guide,

URL: http://developer.android.com/guide/topics/ui/declaring-layout.html

Accessed on 10 May 2015.

[15] BLE, Android Developer's Guide,

URL: http://developer.android.com/guide/topics/connectivity/bluetooth-le.html

Accessed on 10 May 2015.

[16] Services, Android Developer's Guide,

URL: http://developer.android.com/guide/components/services.html

Accessed on 10 May 2015.

[17] Ole Morten, Forum Answer, Nordic Developer Zone,

URL: https://devzone.nordicsemi.com/index.php/what-is-a-client-and-server-in-ble

Accessed on 10 May 2015.

[18] Bluetooth SIG, Technology Overview-GATT,

URL: https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx

Accessed on 01 May 2015.

[19] IRDA, Technology Details

URL: http://www.irda.org/displaycommon.cfm?an=1&subarticlenbr=14

Accessed on 05 May 2015.

[20] About Infrared

URL: http://www.answers.com/topic/infrared

Accessed on 07 May 2015.

[21] NFC Usage

URL: http://nfc-forum.org/what-is-nfc/what-it-does/

Accessed on 27 April 2015.

[22] ZigBee Developer Guide

URL: http://www.zigbee.org/zigbee-for-developers/

Accessed on 07 May 2015.

[23] All about 3D Printing

URL: http://3dprinting.com/what-is-3d-printing/

Accessed on 10 May 2015.

[24] KiCad Developer Guide

URL: http://www.kicad-pcb.org/display/KICAD/KiCad+Documentation

Accessed on 10 May 2015.

[25] Apple Inc., About Core Bluetooth

URL:

https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/AboutCoreBluetooth/Introduction.html

Accessed on 10 May 2015.